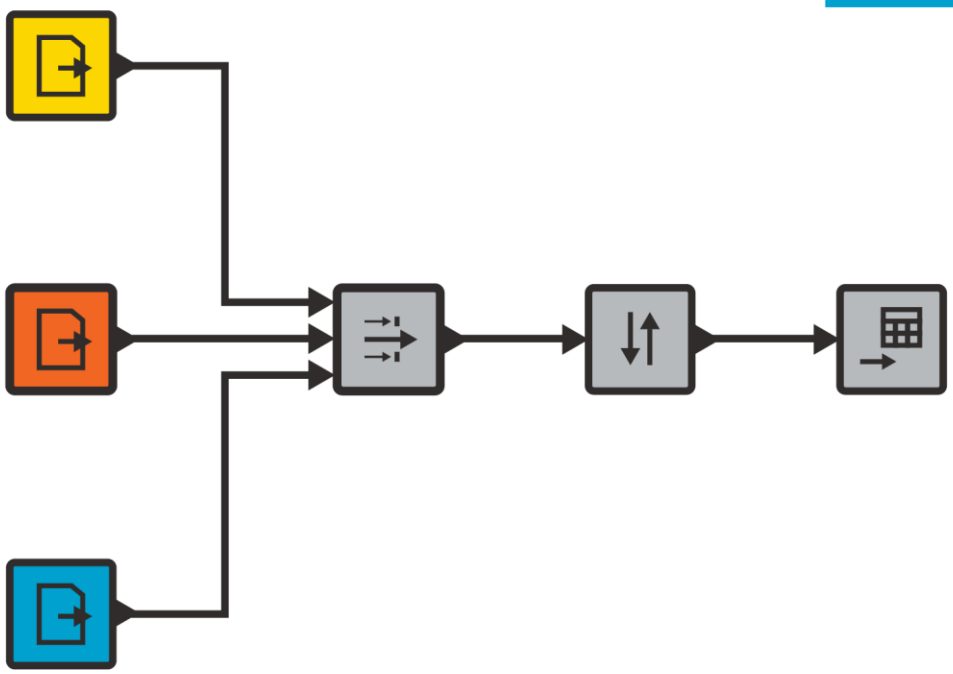
Use Case Collection

Edited by
LADA **RUDNITCKAIA** & ROSARIA **SILIPO**

Will They Blend?

The Connector Collection

KNIME v4.7



Open for Innovation
KNIME

This book has been updated for **KNIME 4.7**.

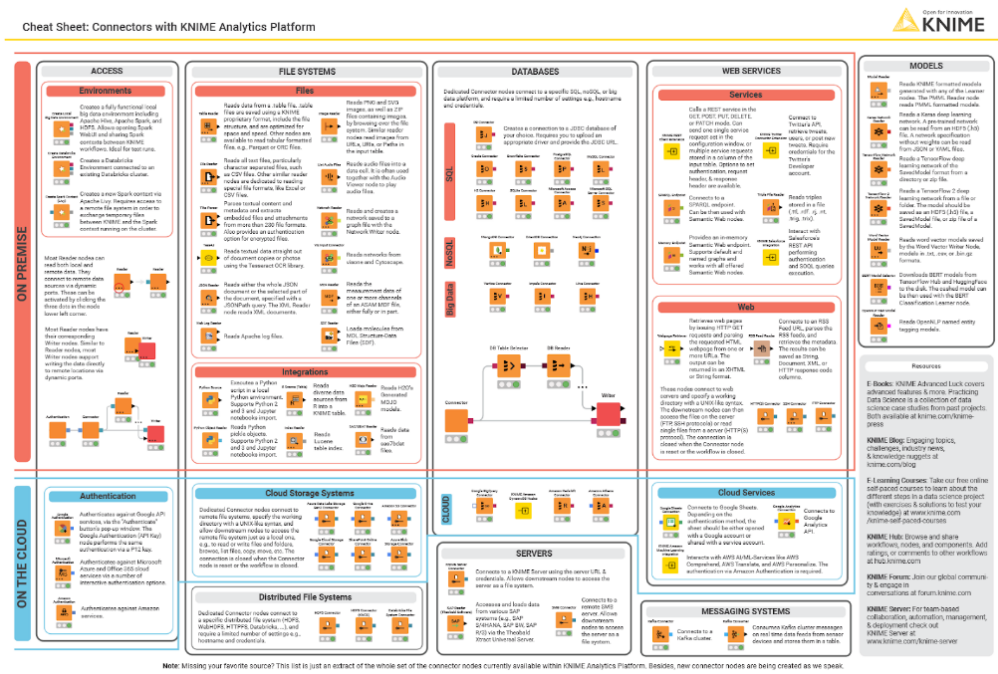
For information regarding permissions and sales, write to:

KNIME Press
Talacker 50
8001 Zurich
Switzerland
knimepress@knime.com

www.knime.com

1, 2, 3, … 100 Connectors

Blend more than 100 data sources smoothly and effectively in a codeless fashion



Download the KNIME Connector cheat sheet for free.

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! We surely need a database, probably more than one, including some traditional SQL and noSQL databases as well as some big data repositories, some databases on premise and some on the cloud. Then, probably, we need to collect data from one or two web services and when we're talking APIs, we need to parse XML and JSON formatted data structures. We might need to integrate a Python script into our workflow as well. Of course, we can't leave out the omnipresent Excel file. Text files also belong to the everyday routine. Spice it up with some vintage data storage

software and you have the picture of where most of your time goes. We could continue – just fill in here the name of your most used data source.

One of the very useful features of KNIME Analytics Platform and its extensions is its versatility to connect easily to a wide variety of data sources throughout a very large number of Connector nodes. Connector nodes are, as the name says, connecting to data sources, each one is specialized in connecting to its own data source. They are straightforward to use, since only a minimal subset of settings is exposed in their configuration dialog. At the same time, they are very flexible, due to their optional configuration that allows for example to add as many diverse input connections as needed.

We will see many of those connectors and many of their features in the course of this book. Many of them are summarized in the Connector cheat sheet, which is downloadable for free from the KNIME web site. In this introduction, though, I would like to give you an overview of the most frequently used connector nodes. Since for obvious space reasons, I cannot describe all of them one by one, I would like to give you a taste of the marvelous world of KNIME Connectors by exploring the Connector cheat sheet.

Let's start from the left: Reading Files. Reading files is one of the most common data access tasks. The **File Reader** node can read various text files, such as **CSV** and **TXT**, and automatically guess the structure of your files. For many special formats, such as **Excel**, there are dedicated reader nodes providing configuration options that are specific to each one.

File formats such as **JSON** and **XML** have dedicated reader nodes. In addition, KNIME provides additional nodes and features to extract their contents in an easy and convenient way. **PDF** files, **Word** documents, and many other formats are covered by the **Tika Parser** node. KNIME is also capable of reading images, audio files and networks.

With the many technologies and services allowing you to store information, it's unlikely that all your data is available locally. KNIME offers an easy option to connect your reader nodes to remote file systems. To activate a file system connection, click the three dots in the bottom left corner of a reader node. KNIME provides connectors for **cloud storage systems**, such as **Azure Data Lake Gen2**, **Google Drive**, **Amazon S3**, and **Microsoft SharePoint**.

There are also connectors for **distributed file systems**, such as **HDFS** and **Databricks**.

To connect to these systems, the corresponding authentication node needs to be used. For instance, the SharePoint Online Connector node requires a **Microsoft Authentication node**. You can find all authentication nodes in the lower left corner of the cheat sheet.

KNIME offers integrations with various other tools like **Python**, **R**, **H2O**, and **SAS**.

Python and R Scripts can be read and executed via the corresponding Source nodes. The parent extensions come with large sets of nodes, such as a Python Object Reader or a Python View node.

Databases are another common way of storing and accessing data. There are dedicated connector nodes for common **SQL** databases, such as **Oracle**, **Snowflake**, **PostgreSQL**, and **MySQL**. For **NoSQL** databases, there are connector nodes to sources such as **MongoDB** and **Neo4J**. For **big data** technologies, we have **Hive** and **Impala**. Connectors to **cloud-based databases** such as **Google BigQuery**, **Amazon Redshift**, and **Amazon Athena** are available as well.

KNIME also offers dedicated nodes to perform **in-database** processing. These nodes allow non-experts to build SQL statements in a visual way. The **DB Reader** node, placed at the end of a sequence of DB nodes, pulls the data from the database into KNIME based on the SQL statement created by the previous nodes. Alternatively, some nodes, like the **DB SQL Executor**, allow the SQL query to run directly on the database, without the data leaving it. This approach exploits the computational power of the database system to process the data.

Web Services can be called by the nodes of KNIME's REST Client extension. **Semantic Web** resources, such as SPARQL endpoints, in-memory endpoints, and Triple files, can be accessed via their corresponding nodes. Additionally, there are various extensions for external web services, such as **Twitter** or **Salesforce**. To retrieve contents of a website or an RSS feed, the **Webpage Retriever** or the **RSS Feed Reader** nodes can be used, respectively. The **HTTP(S)**, **SSH**, and **FTP** Connector nodes allow you to connect to servers via the specified protocol.

Cloud providers such as Amazon, Microsoft, and Google do not only provide cloud storage systems. They also have their own services and solutions for data analysis and data science. KNIME offers integrations with services such as **Google Sheets**, **Google Analytics**, and **Amazon's AI/ML Services** like Comprehend, Translate, and Personalize. Additionally, there are reader nodes for various machine learning model types, such as **KNIME models**, **PMML** formatted models, and **Keras** & **TensorFlow** networks.

Last but not least, KNIME provides a set of nodes to create big data environments or contexts. The **Create Local Big Data Environment** node creates locally a fully functional environment including **Apache Hive**, **Apache Spark**, and **HDFS**. It's ideal for testing your application before deploying it into the real cluster.

With this last group of nodes, we have come back to the connector nodes in the top left corner of the cheat sheet, after zooming in on simple files, cloud files, databases, big data repositories, web services, cloud services, cloud storage systems, and authentication.

Julian Bunzel

Table of Contents

GPT-3 Meets StackOverflow

Who is better at answering: Humans or Machines?

Authors: Anil Özer, Roberto Cadili, KNIME

Workflow on KNIME Community Hub: GPT-3 meets StackOverflow

The Challenge

In a world where technology is constantly evolving, state-of-the-art large language models (LLMs) for human-like text generation have evolved as the undisputed game-changers in the fields of analytics. In late 2022, one such model, ChatGPT, took the data community by storm becoming the go-to solution for whoever seeks answers to all sorts of questions. Whether it's a homework assignment, a code snippet, or just a curious mind, ChatGPT and similar cutting-edge LLMs can provide fairly accurate, comprehensive and well-argued answers in a matter of seconds. With their ability to understand and respond to natural languages, how can we use these models to our advantage and find answers to popular questions on the internet? Can they provide better answers than humans?

To find out, we will put KNIME Analytics Platform to use by building a workflow that utilizes the power of GPT-3, and more specifically its “text-davinci-003” model, to answer the top 10 most frequently asked questions on StackOverflow.

For those who are not familiar with either of today’s players, **GPT-3** stands for Generative Pre-trained Transformer version 3 and is a state-of-the-art large autoregressive language model (175 billion parameters) with a decoder-only transformer network. It was released in 2020 by OpenAI and uses deep learning to understand and generate human-like responses in natural language. GPT-3 was trained on an extremely vast text dataset from multiple sources, including Wikipedia and books. Given an initial text prompt, it can produce text that continues the prompt on a wide range of topics, making it a versatile tool for many industries.

StackOverflow is a question-and-answer forum focused on programming and software development. Users can ask and answer questions, use tags, and express appreciation by upvoting the posted entry. By means of vote counts (both for questions and answers), users can also determine the popularity and relevance of each post.

Usually, the most-highly upvoted answer is considered to be the right one. Additionally, users whose answers are often upvoted receive status badges and can build their reputation in the community.

Both these sources can be accessed and interacted with from their respective REST APIs, using mostly GET or POST methods. To connect to REST-ful web services using KNIME Analytics Platform, we download and install the KNIME REST Client Extension from the KNIME Community Hub.

Topic. Getting and Posting API requests to exchange information with REST-ful web services.

Challenge. Compare how humans answer questions vs. a state-of-the-art LLM.

Access Mode. KNIME REST Client Extension, KNIME JSON-Processing.

The Experiment

The first API that we call is the StackOverflow API, powered by StackExchange. It allows retrieving questions sorted in descending order by the number of votes and from a user-defined date of posting. We can use this API to get a wealth of information about popular questions and answers on StackOverflow, including title, body, tags, score, and more. The second API is the OpenAI API. It allows accessing GPT-3's "text-davinci-003" model for text completion and generation. "Text-davinci-003" is the most powerful model version currently available (as of Feb 2023), and it includes training data up until June 2021.

Questions from StackOverflow

In the lower branch of our KNIME workflow, we start off by retrieving the top 10 questions by total number of votes from StackOverflow.

GET Request

We issue an HTTP GET request to the API using the GET Request node. Conveniently, the StackOverflow API does not require any API keys to retrieve questions or answers from the database. In order to send a GET Request, you need:

1. A valid URL with specifications.
2. The KNIME REST Client Extension.

The URL specification may vary according to what we are interested in. For our experiment, we use the following URL:

https://api.stackexchange.com/2.3/questions?oder=desc&min=1000&sort=votes&fromdate=946684800&site=stackoverflow

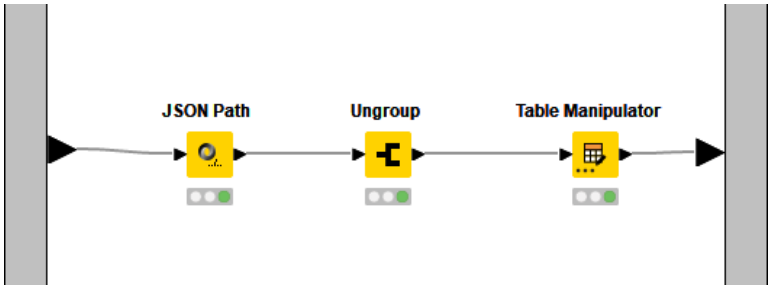
Let's dive into each part of the URL:

- **2.3** denotes that we use the most recent 2.3 version of the API.
- **oder=desc** denotes that questions will be sorted in descending order.
- **min=1000** denotes that only questions with at least 1000 votes will be included.
- **sort=votes** denotes that questions will be sorted by the number of votes.
- **fromdate=946684800** denotes the Unix timestamp for January 1st, 2000 at 12 am so we represent this specific point in time as a single number. We start our query from this point onwards.
- **site=stackoverflow** denotes that questions will come from StackOverflow and not another StackExchange website.

If the GET request is issued successfully and without errors (code status: 200), the API returns a response in JSON format.

Parse JSON Response

Responses in JSON format have a well-defined structure, which makes it easy to interpret and parse useful pieces of information. We can preview and extract relevant values from the JSON response using the JSON Path node. These values are added as collection queries, converting the results into lists of different values. The lists are then disaggregated into several rows using the Ungroup node. Finally, we use the Table Manipulator node to retain columns containing the username, avatar image and profile URL of the person who asked the question, the question text and URL, the number of votes, and the ID of the most upvoted answer for further processing.



We wrap the nodes to parse the JSON response in the "Processing" metanode.

Note. We use this combination of JSON Path, Ungroup and Table Manipulator nodes several times in our workflow to process the JSON response of the APIs from different requests.

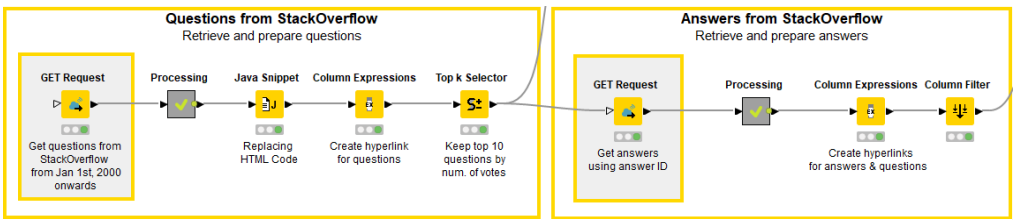
Process Texts and Keep Top 10

Before we can feed questions to GPT-3, we need to perform some text processing. Questions on StackOverflow often include special characters that are not straightforwardly recognized and converted in the JSON response. As a result, the retrieved question text may contain HTML entity numbers instead of the actual characters. To convert HTML entity numbers into their equivalent characters, we use a line of Java code in the Java Snippet node.

Next, the Column Expressions node converts some string values into integers and embeds links for specific columns. Finally, the Top k Selector node is used to filter the top 10 questions sorted in descending order by the number of votes.

Answers from StackOverflow

To be able to compare human vs. machine answers, we also need to get answers from StackOverflow. To do that, we build a URL that contains the answer ID (we obtained it when we retrieved questions) and issue a new GET request to the StackOverflow API.



We use the GET Request node to retrieve questions, answers, and the corresponding metadata from the StackOverflow API.

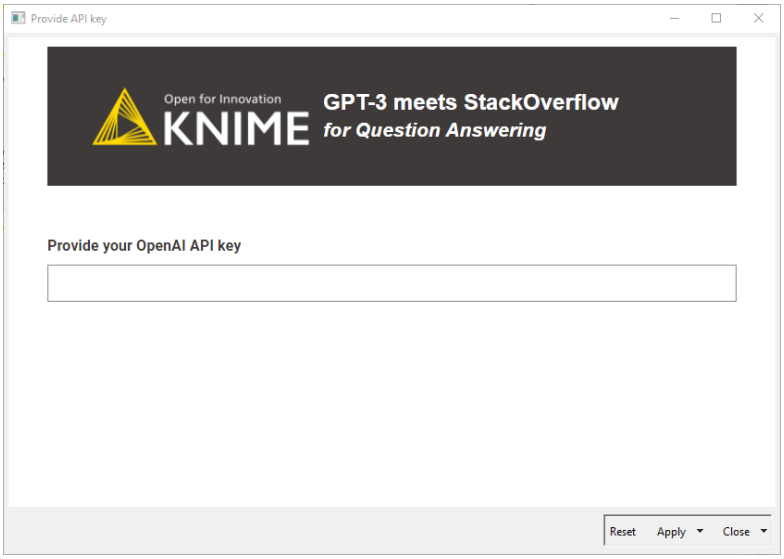
In this case, we are not interested in the actual answer text, as this can be very verbose. Rather, we want to get the direct link to the most upvoted human answer for each retrieved question. We parse the JSON response in the same way we did above and additionally extract the username, avatar image and profile URL of the person who provided the answer, as well as the number of votes the answer received.

Answers from GPT-3

Now that we have the top 10 questions in StackOverflow, we can use the OpenAI API to feed them to GPT-3 and let “text-davinci-003” generate answers for us.

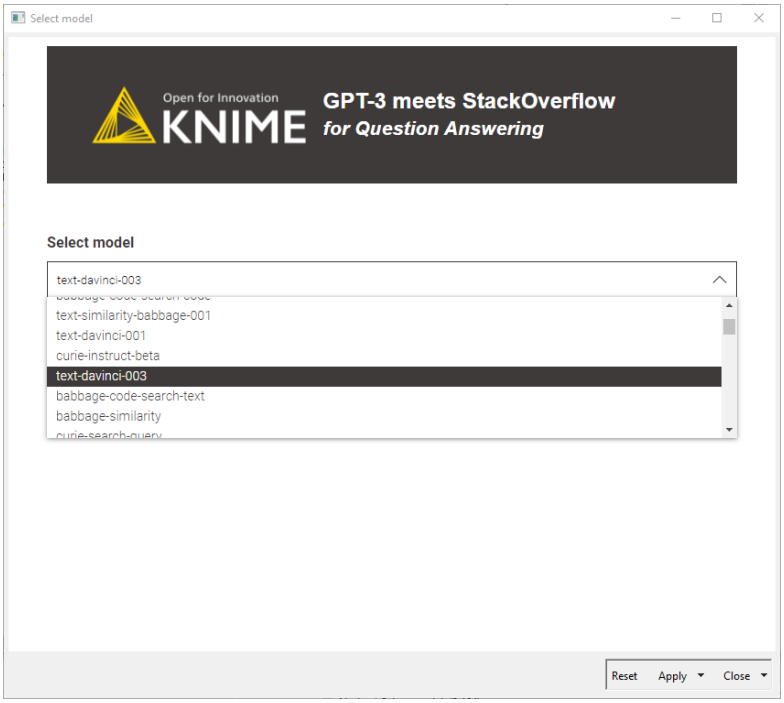
Authenticate and Select Model

To authenticate with the OpenAI API, API keys are utilized. An account at https://openai.com/api/ is required to get access to API keys. When you are logged into your account, visit the “API keys” page. Once you get your API key(s) from there, you can simply enter it in the “Provide API key” component and select “Close & Apply”.



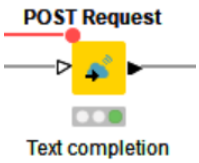
Input your API keys from OpenAI.

The OpenAI API offers the possibility to choose among different models, each with different capabilities, accuracy, professing speed and pricing. We choose the most powerful model that is currently available: “text-davinci-003”.



Select the model to use. Some may be more appropriate than others according to the task.

POST Request



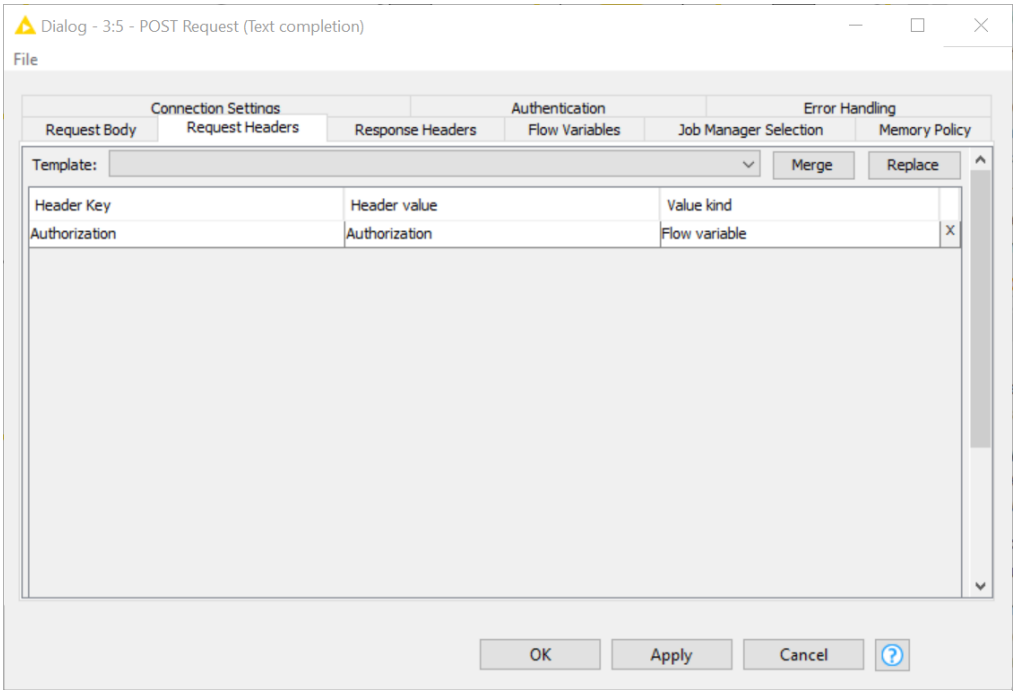
To interact with "text-davinci-003" and let it generate answers starting from a prompt, i.e., questions, we need to issue a HTTP POST request to the API using the POST Request node. To do that, we need:

1. A valid URL with specifications
2. The KNIME REST Client Extension
3. Authentication with an API key
4. A request body in JSON format.

The POST Request node requires a constant URL that we use to access the API. Since we want the model to generate text whenever a prompt is provided, the URL we need is the following:

https://api.openai.com/v1/completions

Next, we need to make sure that our API key is fed into the POST Request node to enable access to the web service. To do that, we pass the API key as a flow variable in the "Request Headers" tab.

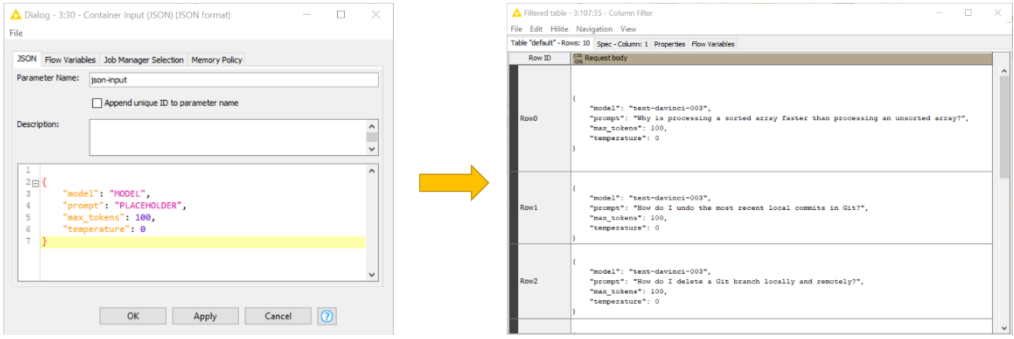


Pass the API key for authentication in the "Request Headers" tab.

Finally, we need to specify a request body in JSON format in the "Request Body" tab. Because the model selection and prompt value change according to the user preference and question, respectively, we can't use a static body. Rather, we pass a table column that contains a custom request body for each question prompt. To do this, we use the Container Input (JSON) node to create the initial structure of the JSON body. There, we add two placeholders – one for the questions and the other for the model, we set the max amount of tokens the answer can have to 100, and specify $temperature = 0$ to ensure that the outputs are mostly deterministic.

In the "Question to JSON" metanode, we merge the Container Input (JSON) node with the output from the Top k Selector node by appending their columns. Missing values are handled by copying the initial JSON request body with the Missing Value node. The String Manipulation node is then used to replace the placeholders with the actual question and model selection in a new column. Finally, the String to JSON node converts this new column into the JSON format that will be used by the POST request. The body of our POST Request is ready and can be prompted to "text-davinci-003".

As with prior API requests, the responses are received in JSON format and processed using the "Processing" metanode. The final step is to merge the table of answers (and questions) from StackOverflow and from "text-davinci-003". This combined table is then used as input to initiate an interactive Data App.

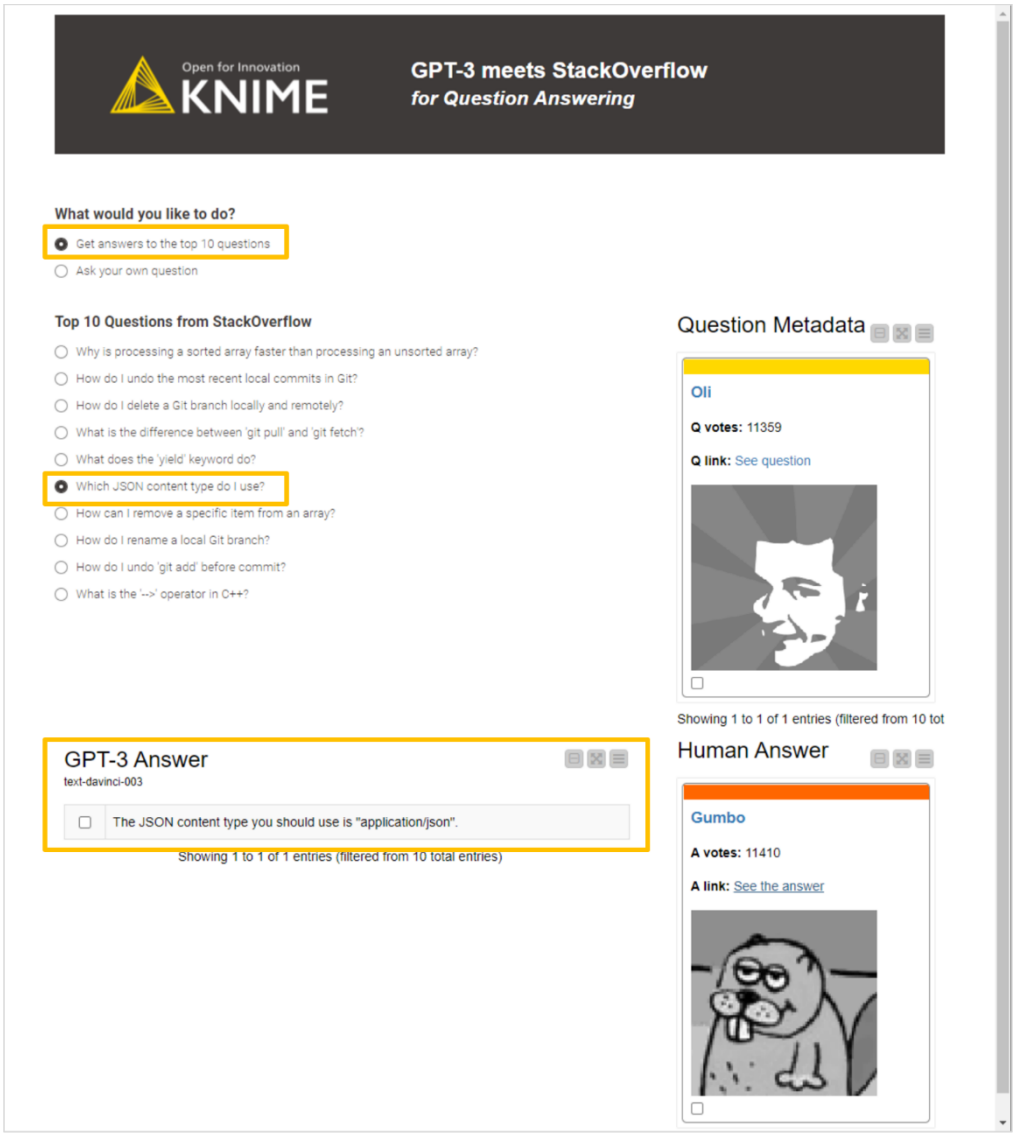


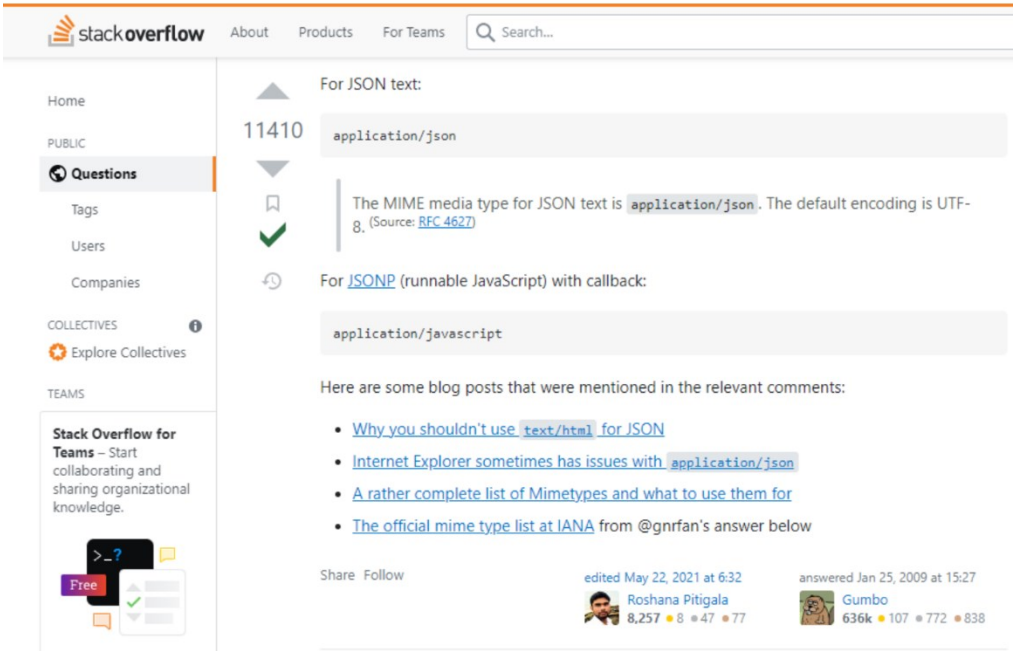
From the general JSON request body (left) to the custom JSON request body that will be prompted to "text-davinci-003" (right).

The Results

We aim to provide users with a range of options for interaction in a KNIME Data App. The first option allows users to explore the top 10 questions, viewing metainformation such as the questioner, the number of votes received, and the question's full content. They can also access the answer provided by "text-davinci-003", as well as the link to the best human answer, the number of upvotes, and the answerer's profile.

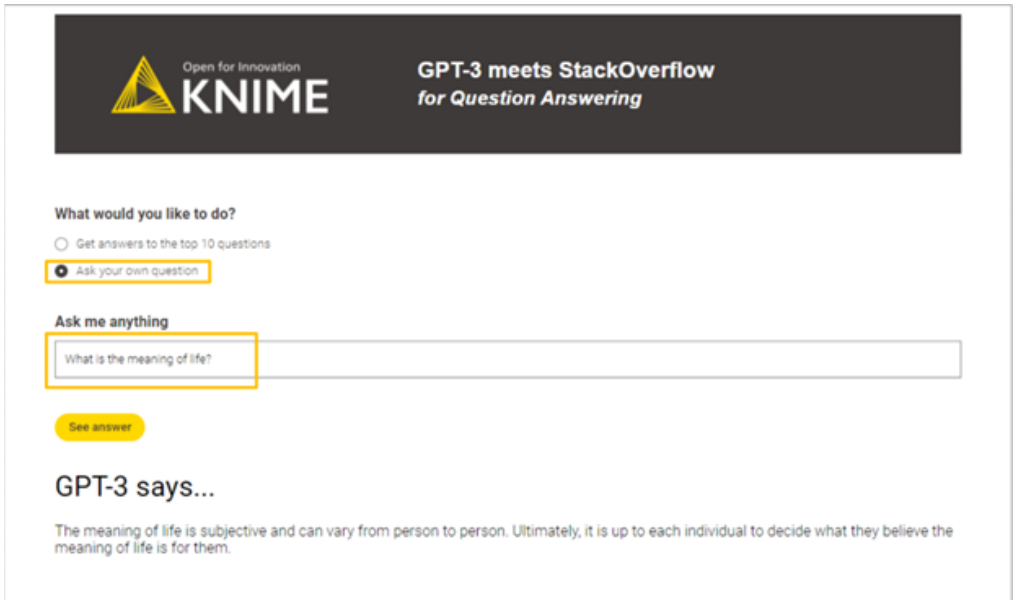
For example, we can see that for the question "Which JSON content type do I use?" GPT-3 provides the correct answer in a very concise and handy way, avoiding the verbosity of human answers. It is also worth considering that GPT-3 was fed solely with the question title. On the other hand, one can argue that the human answer is more exhaustive, including additional resources or explanations which are certainly very valuable too. Hence, according to the user's needs and tweaking GPT-3's max token parameter or prompt provided, comparable results might be achieved.


Open for Innovation
KNIME
GPT-3 meets StackOverflow
for Question Answering
What would you like to do?
Get answers to the top 10 questions
Ask your own question
Top 10 Questions from StackOverflow
Why is processing a sorted array faster than processing an unsorted array?
How do I undo the most recent local commits in Git?
How do I delete a Git branch locally and remotely?
What is the difference between 'git pull' and 'git fetch'?
What does the 'yield' keyword do?
Which JSON content type do I use?
How can I remove a specific item from an array?
How do I rename a local Git branch?
How do I undo 'git add' before commit?
What is the '-->' operator in C++?
Question Metadata
Oli
Q votes: 11359
Q link: See question
Showing 1 to 1 of 1 entries (filtered from 10 tot
GPT-3 Answer
text-davinci-003
The JSON content type you should use is "application/json".
Showing 1 to 1 of 1 entries (filtered from 10 total entries)
Human Answer
Gumbo
A votes: 11410
A link: See the answer



Comparison of GPT-3's answer (top) vs human answer (bottom) for a given question.

The second option enables users to pose their own questions, with the ability to request answers from the model by clicking on the 'See answer' button. For example, we asked GPT-3 about the meaning of life. While the answer does not reveal any mind-blowing truth, it is definitely fair and comparable to what an average human would probably answer.

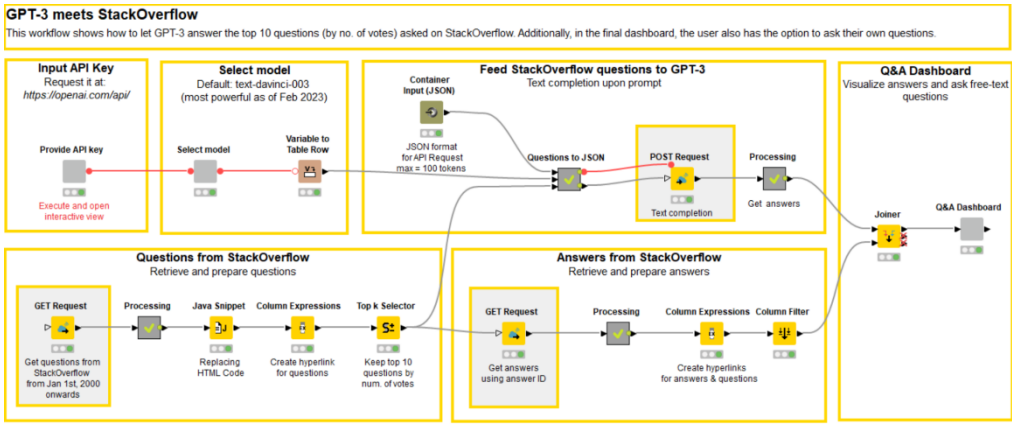


Ask GPT-3 your own questions.

Wrapping up

In this example, we learned how to access, retrieve and blend data and technologies from two APIs via the GET and POST Request nodes with a low code approach. The questions and their corresponding responses procured from these APIs allowed us to effectively visualize them and make a comparison between human and machine generated answers, thereby gaining valuable insights.

Thus, in conclusion, we can say “yes, they blend!”



This workflow, [GPT-3 meets StackOverflow], blends questions from StackOverflow and answers from GPT-3 and is available for download from the KNIME Community Hub.

MDF Meets Apache Kafka

How's the Engine Doing?

Author: Emilio Silvestri, KNIME

Workflow on KNIME Community Hub: MDF Meets Apache Kafka

The Challenge

Let's drive around the world of automotive and experiment with measurement data. Our new engine is undergoing tests and we want to compare its performances with the temperatures that are reached. A very sophisticated and standardized sensor measures the speed of the motor and the turbine and produces an MDF file with the measurements. Another sensor keeps track of the temperature publishing those values on a Kafka Topic. How can we merge and compare these different data?

Let's first introduce our tools.

MDF was originally developed as a proprietary format for the automotive industry. Thanks to its stability and versatility, in 2009 a new ASAM working group released the first standardised version of the ASAM MDF file format, which has since been adopted as the *standard de facto* in the field of measurement and calibration. It allows for easy storing and reading of measurements data as well as any related meta information.

Apache Kafka is an open-source streaming platform able to deal with real time data feeds. Many sensor devices (called *Producer*) can be configured to publish their measurement data regarding a certain *Topic* to a *Kafka Cluster*. The data are then be read by the *Consumers* subscribed to that topic. If you are interested, here is a complete overview of Kafka.

In an ideal world, all our devices would speak the same language and be able to communicate with each other. But if you have ever worked with raw sensor data you probably know that this is not the case: different sampling rates, data formats, ranges, units of measurement… There are many differences within the data that can make this process tricky and unfunny.

Luckily, with KNIME Analytics Platform this becomes child's play!

Topic. Analyze automotive related sensor data

Challenge. Blend sensor measurements in MDF file format and from a Kafka Cluster

Access Mode. KNIME MDF Integration and KNIME Extension for Apache Kafka (Preview)

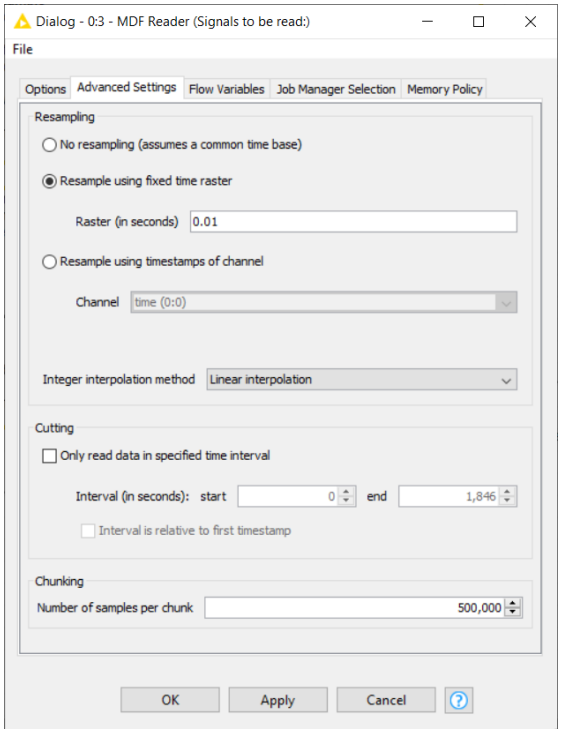
The Experiment

First, let's focus on the MDF measurements. In order to read the *sample.mdf* file attached to the workflow, we use the MDF Reader node. The node comes with the KNIME MDF Integration which you can download from the KNIME Community Hub.

This node is based on the asammdf Python library, therefore the KNIME Python Integration must be set up. Refer to the KNIME Python Integration Installation Guide for more details. The MDF Reader node offers a variety of settings to deal with the MDF file format. In the option tab of the configuration window, select the MDF file (you can use either an absolute or a relative path) and in the Channel Selection menu mark the channels you want to focus on. In this example we will read both of the available channels. MDF file is organized in binary blocks, and a channel is a binary block that stores information about the measured signal and how the signal values are stored. Another important binary block is the data block that contains the signal values.

Move to the Advanced Settings tab and explore further options:

- **Resampling**: measurements from different channels might not have the same sampling rates and offset. This option will do the resampling for us. You can choose the interpolation method - linear or previous value - and the channel's timestamp on which the resampling is performed. Otherwise, you can define your own sampling rate. The temperature data that we are going to read later are sampled every 0.01 seconds. Therefore, let's configure the MDF node to resample at this specific interval as shown in the configuration window below.
- **Cutting**: only the data within the specified time interval are read.
- **Chunking**: only reads the specified amount of measurements. This is useful when the file does not completely fit into the main memory.

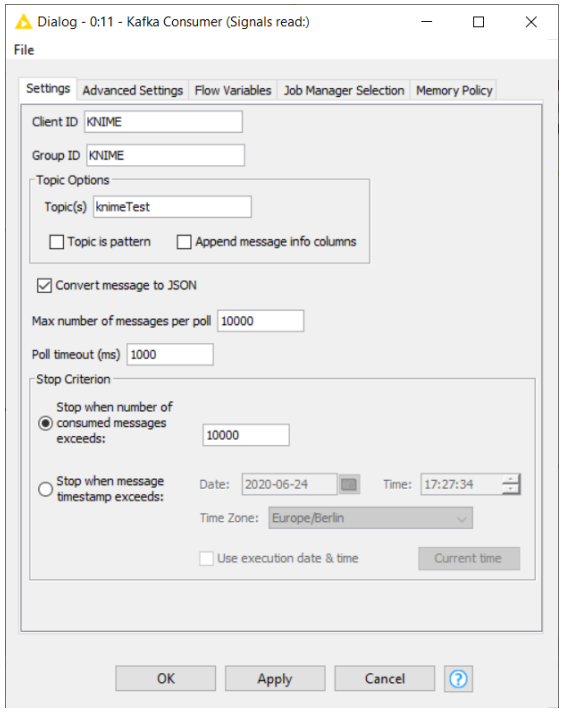


Advanced settings of the MDF Reader node.

The second part of our measurements - regarding the temperature of the engine - are sent by the sensor to a Kafka Cluster. KNIME Analytics Platform supports this technology thanks to the KNIME Extension for Apache Kafka (Preview). The Kafka Connector node will establish a connection with the Kafka Cluster. Let's append a Kafka Consumer node to read the data published to the topic "*knimeTest*" as shown in the configuration window below. This node is also configurable to read a maximum number of entries (Kafka calls them *messages*) or stop reading at a custom time.

Note. The temperature data provided with this workflow have been synthetically generated.

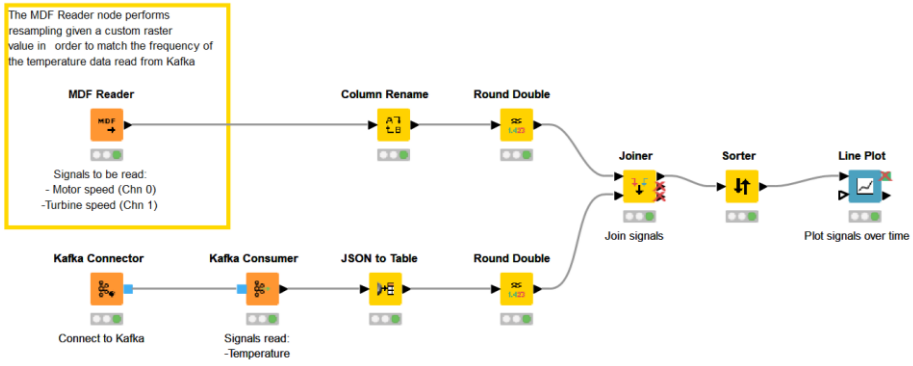
As previously mentioned, these temperature measurements have been recorded at intervals of 0.01 seconds. Since the time offsets match and the MDF Reader node has already performed the resampling of the data...we are ready to blend!



The configuration window of the Kafka Consumer node.

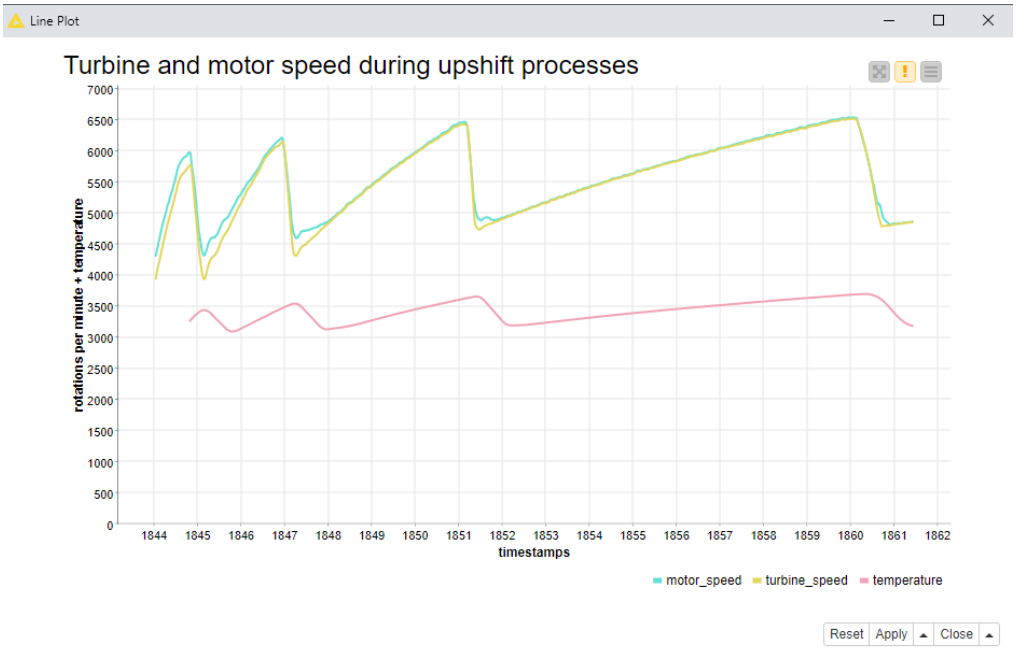
Blending MDF Data with Data from a Kafka Cluster

The Joiner node in the final workflow will merge the data from the two sources according to the time offset value. Please note that because of the resampling, we don't have the exact measurement value for each timestamp but its approximation generated by linear interpolation. The MDF Meets Apache Kafka workflow is available on the KNIME Community Hub for you to download and try out.



Final workflow blending MDF and Apache Kafka measurement data. You can download the MDF Meets Apache Kafka workflow from the KNIME Community Hub

The Line Plot shows the measurements. The green and yellow lines above with more fluctuation show the motor and turbine speed. The red line below shows slight increments of the temperature after the phases of higher motor/turbine speed.



Line plot of the measurements. The values from the different sources have been resampled and joined on the same time offset.

Amazon S3 Meets DynamoDB

Combining AWS Services to Access and Query Data

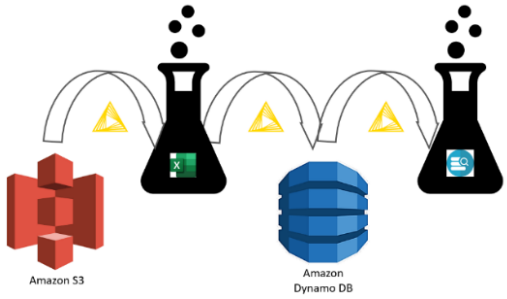
Authors: Scott Fincher, KNIME

Workflow on KNIME Community Hub: Amazon S3 Meets DynamoDB

The Challenge

Today we want to use KNIME Analytics Platform to read and write information using Amazon S3, an object storage service, and then access that data directly to create, populate and interact with Amazon DynamoDB, a NoSQL key-value and document database. With the 4.2 release, KNIME is one of the few tools that allows you to connect to DynamoDB tables using a graphical user interface, without writing any code. All we need are the freely available KNIME Amazon S3 Connector and KNIME Amazon Dynamo DB Nodes extensions, plus an Amazon account with the appropriate credentials.

In our example, we'll show you how to authenticate against Amazon services within KNIME, access data with S3, and then create and load a DynamoDB table. We'll also demonstrate several utility nodes that query against DynamoDB, list and describe available tables, and provide batch operations as needed.



Accessing files on Amazon S3, creating an Amazon DynamoDB table, and manipulating the data directly on DynamoDB. All these steps have dedicated nodes in KNIME Analytics Platform.

Can we blend Amazon S3 data with DynamoDB tables without writing any code? Let's get started and find out!

Topic. Extraction, transformation, and loading (ETL) of US Census data.

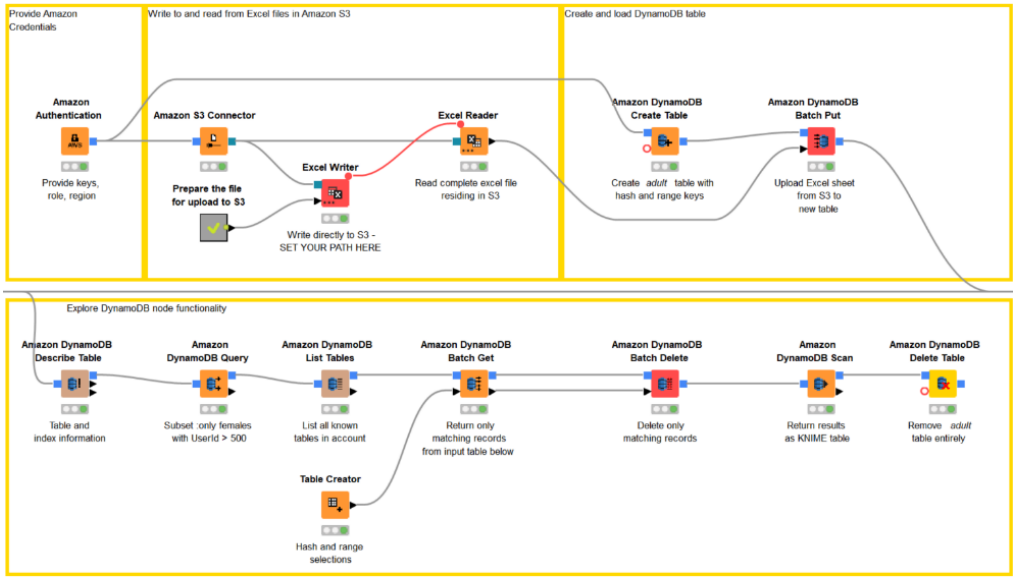
Challenge. Access data on Amazon S3 and load and manipulate it on DynamoDB, code-free.

Access Mode/Integrated Tool. Connect via KNIME file handling features and nodes to Amazon S3 and Amazon DynamoDB.

The Experiment

The workflow shown below visually lays out our ETL process, which will proceed in four parts, each one implemented within a separate yellow annotation box:

1. Amazon authentication
2. Connection and file access on Amazon S3
3. Creation and loading of an Amazon DynamoDB table
4. Exploration of different DynamoDB functions via KNIME nodes



A workflow to access data via S3, load the data into a DynamoDB table, and perform various ETL operations on that table. The Amazon S3 Meets DynamoDB workflow can be downloaded from the KNIME Community Hub.

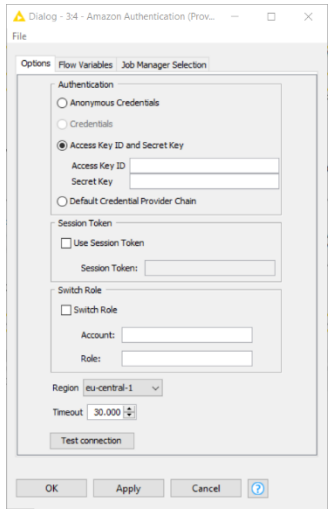
Amazon Authentication

Since most nodes in this workflow will require access to an Amazon account in some fashion, let's start with an Amazon Authentication node.

Here, we provide our Access Key ID and Secret Key, and in addition elect to use the Switch Role option, which requires that we provide an associated account and role as well, granting us additional permissions.

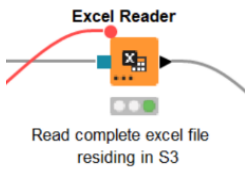
Once we select the appropriate region, we can click the Test Connection button to ensure all the information we've provided is correct.

Note the blue square connection port this node provides – the Amazon connection information that it contains will be propagated downstream to other Amazon nodes, so that we only need to provide this login information once.



The configuration window of the Amazon Authentication node.

Connection and file access Amazon S3

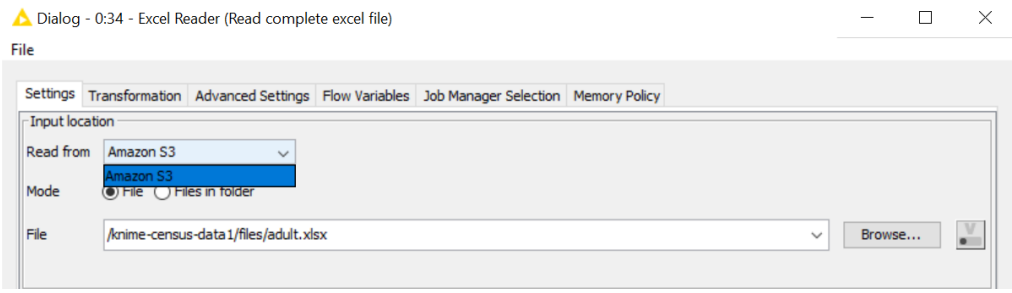


The Excel Reader node to access .xlsx files.

Now, let's connect to S3 and upload some data that we'll eventually transfer into an Amazon DynamoDB table. In this case, we'll use the well known *adult* dataset provided by the UCI Machine Learning Repository. The dataset is based on US Census data and contains information on individuals like gender, marital status, location, and so on. To begin, we'll use an Amazon S3 Connector node. This node will take the login credentials we've already provided and gives us access to an S3 bucket to which we can write data – all we need to provide is a path to a working directory. The output port from this node is a teal colour, which represents an Amazon S3 connection.

We access the adult.xlsx file on S3 with an Excel Reader node using the file handling features introduced in KNIME 4.1. The Excel Reader node can directly access Excel files available on S3 if an Amazon S3 Connection is provided in its optional input port. This feature is enabled by clicking the three dots on the bottom left corner of the node.

Having directly read data from S3, we can now focus on moving that data in DynamoDB for further exploration and manipulation.



Configuration of optional file handling options for direct connection to Amazon S3.

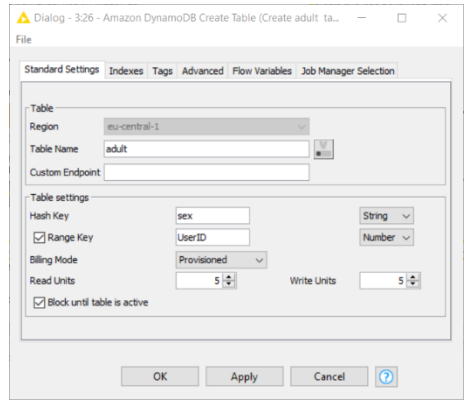
Creation and loading of an Amazon DynamoDB table

The next thing we'd like to do is create an empty DynamoDB table. For that, we need to specify a table name and a primary key that uniquely identifies each item in the table. There are two kinds of primary keys in DynamoDB: simple and composite.

- A *simple key* consists of a partition key (hash attribute) that determines the partition (physical storage internal to DynamoDB) in which the item will be stored. In this case, the partition key should be unique for each item.
- A *composite key* consists of a partition key (hash attribute) and a sort key (range attribute). All items with the same partition key value are stored together and

sorted by the sort key value. In this case, the partition key is possibly not unique, but the sort key value should be unique for the items with the same partition key.

We can create the table using the Amazon DynamoDB Create Table node. In its configuration we are presented with several options to choose a table name, define appropriate hash and range keys (and their associated types), choose a billing mode, and set read and write units.



The configuration window of the Amazon DynamoDB Create Table node.

Here we'll choose a hash key of sex (not unique) and a range key of UserID (unique), being careful to match case with the existing fields in our S3 table. For other choices, we'll stick with the default values.

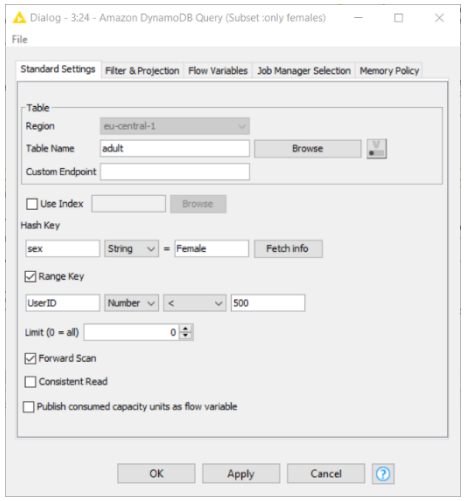
Note that on a separate tab, we also have the ability to add indices and tags for our tables, if desired.

Now that the table is ready, we can use the Amazon DynamoDB Batch Put node to load our data from S3 into it. The configuration allows you to browse existing DynamoDB tables in your account and choose an appropriate batch size.

Exploring different DynamoDB functions via KNIME nodes

With our data now loaded, there are a number of different nodes available in the Amazon DynamoDB extension for querying, exploring, as well as other utility functions:

- Amazon DynamoDB List Tables: Sees what tables are available on our account.
- Amazon DynamoDB Describe Table: Delves into the structure of individual tables. This node has two notable output ports: one that provides detailed information on a table, like its AmazonID, arn, status, size, creation date, and many other features; and another that yields specific information on indices that may exist.
- Amazon DynamoDB Query: Writes direct queries on hash and range keys that we've defined. For instance, in our adult dataset, maybe we're only interested in females with a UserId less than 500. Since we already defined sex as a hash key and UserID as a range key, the configuration dialog of the node makes this very easy:



The configuration window of the Amazon DynamoDB Query node.

We also have the option to provide filter, projections and name/value mapping on an optional tab:

- Amazon DynamoDB Batch Get: Matches particular records in the table. This node takes input from a KNIME table that defines particular combinations of keys I'm interested in – in this case, sex and UserID.
- Amazon DynamoDB Batch Delete: Removes records from my table. The node will accomplish this using a dialog similar to the Amazon DynamoDB Batch Get node.

Finally, after I do some querying on my table, maybe I'd like to read the current state of a DynamoDB table back into KNIME native tabular form for additional processing, or just remove the DynamoDB table if I'm done with it. That's possible with these nodes:

- Amazon DynamoDB Scan: Reads Amazon DynamoDB table into KNIME
- Amazon DynamoDB Delete Table: Deletes tables on the Amazon DynamoDB database

The Results

In the end, it's my choice – I can use either S3 or local data for loading into DynamoDB. Once it's there, I have lots of options for querying and manipulating my dataset directly in DynamoDB, without writing any code. And ultimately, I always have the option to bring my data back into KNIME for more processing, model development, visualization, or deployment.

So when looking at KNIME, S3, and DynamoDB, it's safe to say – yes, they blend!

Theobald Meets SAP HANA

Build a KPI Dashboard

Authors: Maarit Widmann & Andisa Dewi, KNIME

Workflow on KNIME Community Hub: Will They Blend: SAP Theobald Meets SAP Hana

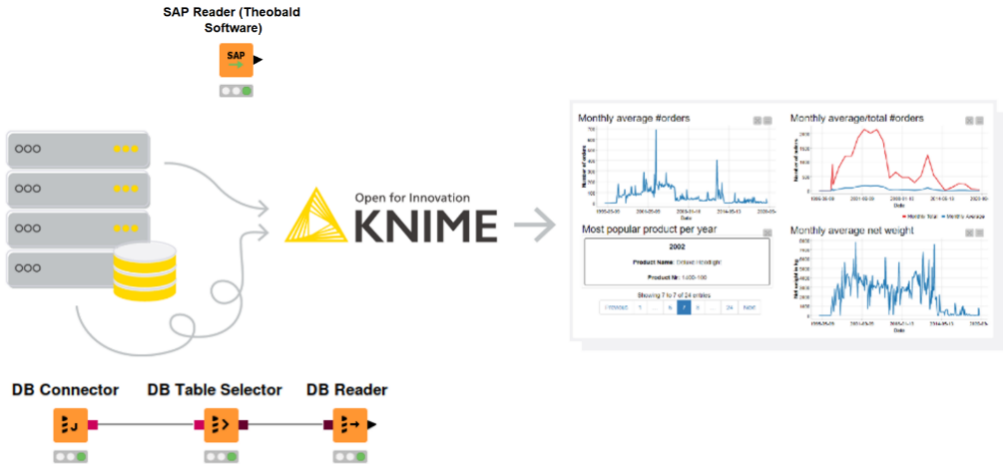
The Challenge

In this challenge we're going to blend data on a SAP system that is accessed in two ways.

1. The legacy way - via the JDBC driver of the database, SAP HANA for example, and
2. The new way - via the Theobald Xtract Universal Server.

The legacy way requires a few steps: registering the JDBC driver of the SAP HANA database in KNIME, connecting to the database with the DB Connector node, selecting a table on the connected database, and reading it into KNIME. Using the SAP integration available from KNIME Analytics Platform version 4.2 forward, you can access and load SAP data into KNIME just by one node called the SAP Reader (Theobald Software).

In our example, we extract KPIs from orders data, and show their development over time. We access the data about the submitted orders the legacy way. The data about the features of the ordered items are available on the Theobald Xtract Universal Server, so we read this data with the SAP Reader (Theobald Software) node. We connect to both systems from within KNIME, join the data on the sales document key column available in both tables, and show the historical development of a few KPIs in an interactive dashboard: Is the number of orders increasing over time? What is the most popular product per year? Let's take a look!



Accessing SAP data via Theobald (new) and via the JDBC driver of the SAP HANA database (legacy) before blending and preprocessing the data and calculating and visualizing KPIs in KNIME Analytics Platform.

Topic. Calculate KPIs of orders/items data and visualize the historical development of the KPIs in an interactive dashboard

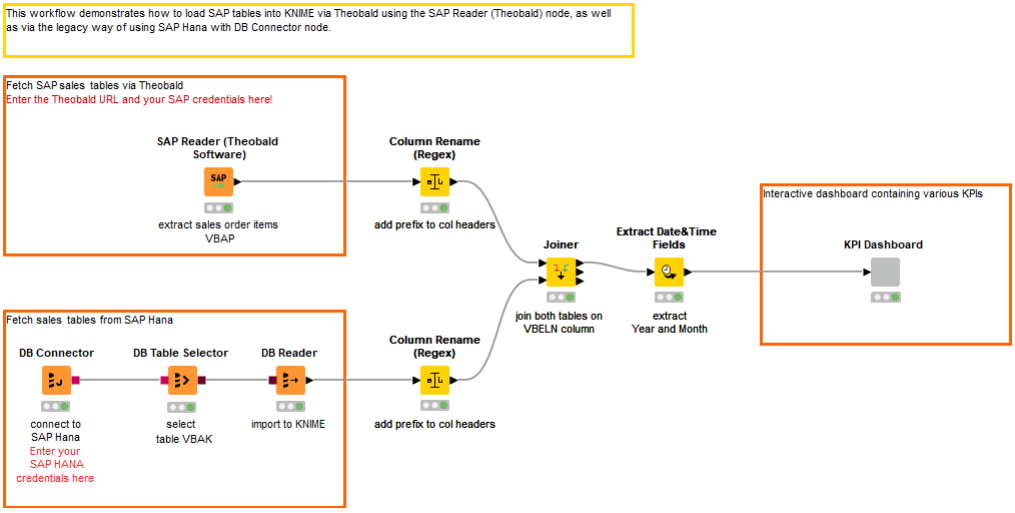
Challenge. Access SAP data via Theobald and via the JDBC driver of the SAP HANA database

Access Mode. Connect to Theobald Xtract Universal Server and to SAP HANA database

Integrated Tool. SAP, Theobald

The Experiment

The workflow in shown below shows the steps in accessing the SAP data via Theobald (top branch) and via the JDBC driver of the database (bottom branch). The data for this experiment are stored in the Sales Document: Item Data and Sales Document: Header Data tables included in SAP ERP. After accessing the data, we join the tables, and extract year and month from the timestamps in order to calculate the KPIs at a meaningful granularity. Next, we calculate four different KPIs: total number of orders per month, average number of orders per month, average net weight of an order in each month, and the most popular product in each year. The KPIs are shown in the interactive view of the KPI Dashboard component. You can download the workflow Will They Blend: SAP Theobald meets SAP HANA from the KNIME Community Hub.



A workflow to access SAP data via Theobald and via the JDBC driver of the SAP HANA database, and to blend and preprocess data before calculating and visualizing KPIs in an interactive dashboard. The Will They Blend: SAP Theobald meets SAP HANA workflow can be downloaded from the KNIME Community Hub.

Accessing Theobald Xtract Universal Server

We want to access the "Sales Document: Item Data" table, which contains detailed information about item orders (each row contains details of a specific item of an order) and is accessible via the Theobald Xtract Universal Server. The server provides a so-called table extraction feature where we can extract specific tables/views from various SAP systems and store them as table extraction queries. The KNIME SAP Reader (Theobald Software) node is able to connect to the given Xtract Universal Server to execute those queries and import the resulting data into KNIME.

1. Open the configuration dialog of the node and enter the URL to the Theobald Xtract Universal Server. Click the *Fetch queries* button to fetch all available extraction queries on the server. We can then select one query from the drop-down list, in our case it is the "Sales Document: Item Data" table. Note that it is necessary to provide SAP credentials in the authentication section if the selected query is connected to a protected SAP system.
2. Executing the node will execute the selected query on the Xtract Universal server and imports the data into a KNIME table.

Accessing SAP HANA

We want to access the "The Sales Document: Header Data" table that contains information about the submitted orders and is available on the SAP HANA database

on a locally running server. We can access the database, like any other JDBC compliant database that doesn't have a dedicated connector node, with the DB Connector node. In the configuration dialog of the DB Connector node, we can select the JDBC driver of an arbitrary database in the Driver Name field. In order to make our preferred database SAP HANA show in the Driver Name menu, we need to register its JDBC driver first.

1. To register a JDBC driver in KNIME, go to File → Preferences → KNIME → Database. The driver (.jar file) is installed as part of the SAP HANA client installation. To find where the JDBC driver is located, please check the SAP HANA documentation. Then we can add it to KNIME by following the steps described here in the KNIME Database Extension Guide.
2. To register a JDBC driver in KNIME, go to File → Preferences → KNIME → Database. The driver (.jar file) is installed as part of the SAP HANA client installation. To find where the JDBC driver is located, please check the SAP HANA documentation. Next, we can add it to KNIME by following the steps described here in the KNIME Database Extension Guide.
3. After registering the JDBC driver, open again the configuration dialog of the DB Connector node, select the newly registered JDBC driver in the menu, for example, sap: [ID: sap_id], and specify the database URL, for example, jdbc:sap://localhost:39015. Also provide the credentials with one of the authentication methods.
4. The connection to the SAP HANA database is now created. Continue with the DB Table Selector node to select a table on the database and the DB Reader node to read the data into a KNIME table.

Blending data and calculating the KPIs

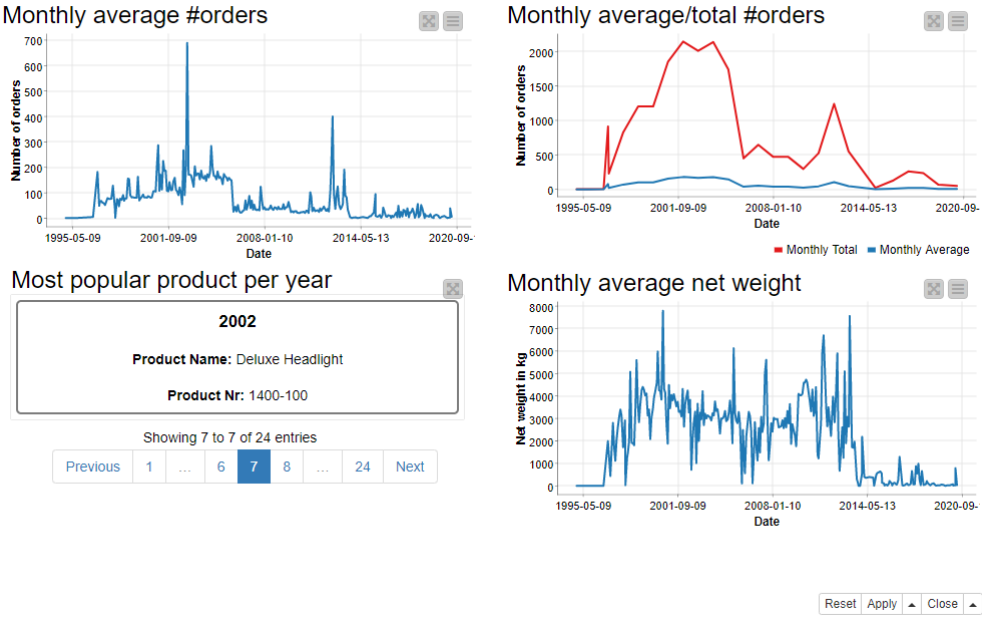
After accessing the two tables, we join them on the sales document key (VBELN column) and get a table that contains information on both the submitted orders and the items included in each order. Since the current granularity of the data is daily, and the time range of the data reaches from January 1997 to May 2020, we aggregate the data at a monthly level before calculating the KPIs.

The Results

The interactive view output of the KPI Dashboard component shown below visualizes the KPIs. In the line plot in the top left corner, we can see that the average number of orders per month was the highest at the beginning of 2000, and since 2014 it stagnates at a relatively low level. Yet in the past there were some quieter periods followed by periods with more orders, for example, the low around 2008 followed by a peak around

2013. We can find a similar kind of pattern in the line plot in the top right corner that shows in addition the total number of orders per month.

In the tile view in the bottom left corner, we can browse through the most popular products for each year. And finally, in the line plot in the bottom right corner, we can see that the ordered items have become lighter over time, or the orders contain less items than before and have therefore less weight. An especially remarkable decrease in the average weight of an order happened around 2014.



Interactive dashboard visualizing the KPIs that were calculated after accessing and blending the orders/items data available as SAP tables.

Do they, or don't they?

In the dashboard shown above, the product names and weights come from the “Sales Document: Item Data” table, accessed via Theobald Xtract Universal Server, whereas the order counts come from the “Sales Document: Header Data” table, accessed via the JDBC driver of an SAP HANA database.

All this information can be visualized in one dashboard, so yes, they do blend!

Microsoft SharePoint Meets Google Cloud Storage

Generate Invoice PDF Reports

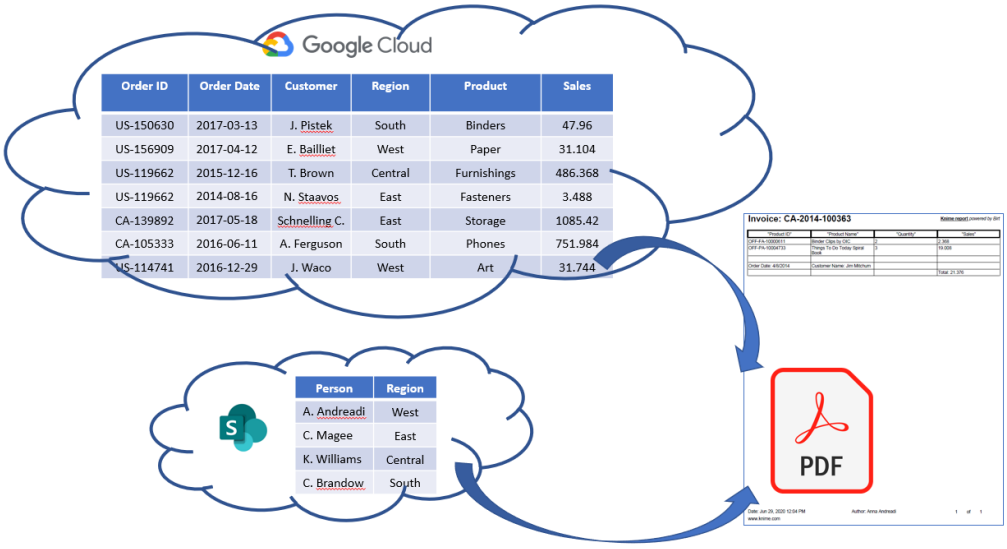
Authors: Maarit Widmann, KNIME

Workflow on KNIME Community Hub: Microsoft SharePoint Meets Google Cloud Storage

The Challenge

In this challenge, we're going to help the accountants of a superstore company, and generate invoices automatically based on orders information. Each invoice contains the order id, order date, customer name, ordered products and their costs, and possibly other information.

The company stores the huge amounts of orders data on Google Cloud Storage. We need to access, filter, and put the data into a shape that fits an invoice report. After that, we export the report as a pdf file. However, an official document from person to person is always nicer, isn't it? Therefore, we want to use the company internal data about the customer contact partners and their regions of responsibility, blend this information with the region where the order was submitted, and add the person's name in the footer of the pdf. The company internal data resides in the collaboration space of the company, Microsoft SharePoint.



Blending data from Google Cloud Storage and SharePoint Online, and exporting the results into a pdf file. In our example we automatically generate invoice reports based on orders data and company internal data.

Topic. Generate invoice reports as pdfs

Challenge. Blend data from Google Cloud Storage and Microsoft SharePoint to access all required information for an invoice report. Export the invoice report as a pdf file.

Access Mode. CSV files

Integrated Tool. Microsoft SharePoint, Google Cloud Storage

The Experiment

The superstore company sells furniture, technology, and office supplies in North America for both corporates and consumers. For each order we have the information regarding the order id, order date, customer name and location, sales amount, and profit. In addition, we have company internal data containing the names of the responsible persons for the West/East/Central/South regions. The original Sample-Superstore.xls file is provided by Tableau.

Step-by-step guide to accessing Google Cloud Storage

1. We start by accessing the order data available on Google Cloud Storage, which is one of the services in Google Cloud Platform. We access the data with the Google Cloud Storage Connector and Google Authentication (API Key) nodes.
2. In the configuration dialog of the Google Authentication (API Key) node, we provide the service *account email* and *P12 key file*. We can get these by following the steps described in detail in the Tutorial: Importing Bike Data from Google BigQuery blog post: signing in with a Google account, creating a project, creating a service account, and downloading the API key in P12 format.

   > ***Note.*** Make sure to assign the Storage Admin role to the service account that you use to access Google Cloud Storage from KNIME analytics Platform.

3. Further down in the configuration dialog of the Google Authentication (API Key) node, in the scopes field, we select the Google services, such as Cloud Storage, Drive, and Sheets, which are granted for this connection. We select Google Cloud Storage (Read) in the dropdown menu, and click the Add button. When we do this, we see the line https://www.googleapis.com/auth/devstorage.read_only in the scopes field.
4. Next, we access the order data with the Google Cloud Storage Connector node. In the configuration dialog, we provide the project id, and select a working directory where the subsequent Reader nodes in the workflow start the browsing for files. The project id is shown in the top left corner of the Google Cloud Platform dashboard. Our data reside in the "Superstore" bucket in the "Orders" folder, so we write "/Superstore/Orders" in the working directory field and click OK.
5. Now, we're ready to read the data into KNIME! We simply use the CSV Reader node, and connect the Google Cloud Storage Connector node to its optional File System Connection input port. In the configuration dialog of the CSV Reader node, we click the Browse button, and select the Sample-Superstore.csv file.

Step-by-step guide to accessing data on SharePoint Online

1. In this second step, we access the company internal data on Microsoft SharePoint, the collaboration space of the company. We can access SharePoint Online, one of the Microsoft 365 cloud services, with the Microsoft Authentication and SharePoint Online Connector nodes.
2. In the configuration dialog of the Microsoft Authentication node, we have two options for authentication: interactive authentication or username and password.

We select “Interactive authentication” in the dropdown menu and click Login. A dialog opens asking for the Microsoft account credentials and security consent before proceeding. Agreeing creates the connection, confirmed by a green “logged in” message next to the Login button in the configuration dialog. Alternatively, we could select Username/password authentication in the dropdown menu and write the Microsoft account credentials in the fields that activate.

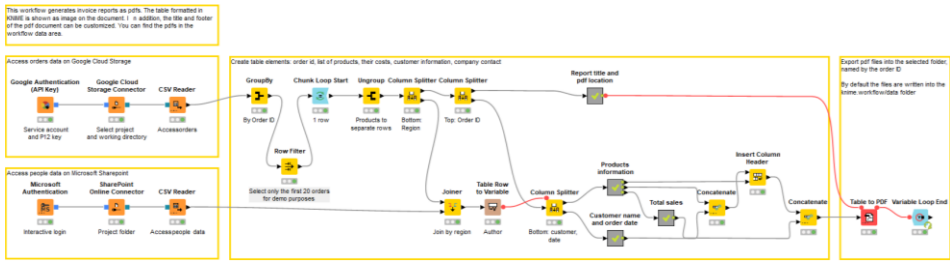
3. Further below in the configuration dialog we can define the reach of the authentication. The “memory” setting resets the connection when the workflow closes. The “file” setting enables using the same connection in multiple workflows. The “node” setting enables closing and opening the workflow without recreating the connection.
4. Finally, we define the access rights for the connection. For this experiment, the SharePoint files (Read) access is enough.
5. Next, we access the data in the collaboration space with the SharePoint Online Connector node. In its configuration dialog, we can directly access the “root site”, the uppermost folder level in the company’s SharePoint site, or the Web URL, such as superstore.sharepoint.com, or if the admin consent was given in the authentication step, we can also select a group site directly.
6. We can select a working directory by clicking the Browse button next to the Working directory field, and the subsequent Reader nodes will browse for folders and files starting from this folder level.
7. Finally, we use the CSV Reader node to read the Superstore-People.csv file containing the names of the responsible persons for different regions.

Blending, wrangling, and exporting data into a pdf file

The data blending and preprocessing steps are shown below in the Microsoft SharePoint Meets Google Cloud Storage workflow. The workflow can be downloaded from the KNIME Community Hub. After accessing the data, we start a chunk loop to handle only one order at a time. Next, we reshape the data for each order so that

1. Each ordered product appears in a separate row
2. The customer’s name, order date, and total value appears at the bottom of the table
3. The order ID appears in the invoice report title

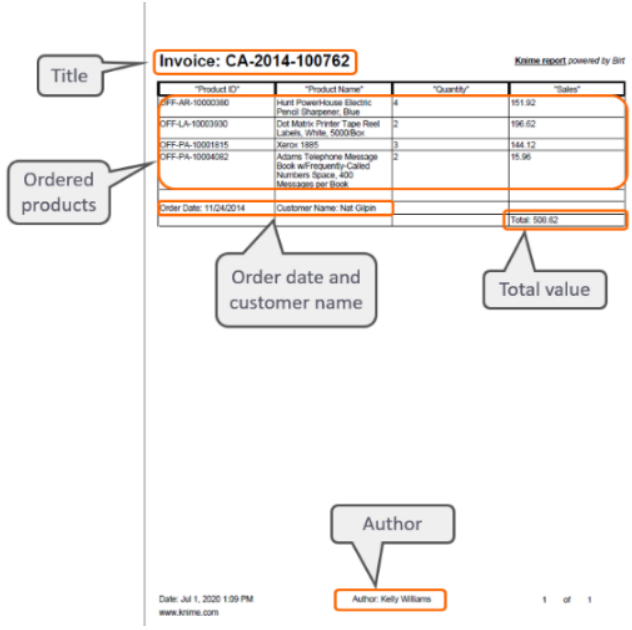
4. The employee's name appears in the footer



Blending data from Google Cloud Storage and Microsoft SharePoint, preprocessing the table into a shape that fits an invoice report, and exporting the table into a pdf with a custom title and footer. The procedure is automatically repeated for all orders in the data. The Microsoft SharePoint Meets Google Cloud Storage workflow is available on the KNIME Community Hub.

Therefore, we join the information about the employee's name to the order data based on the region where the order was submitted. We concatenate the rows for the products, customer name, and the total value, because this information is shown in the invoice table. We combine the string "Invoice:" with the order id to customize the title of the invoice report. Finally, we provide the invoice table as the data input, and the employee's name and report title as flow variables for the Table to PDF node.

The Table to PDF node generates a pdf, where the input table appears as an image, with a custom title and author. The pdfs are exported into the location specified in the configuration dialog - whatever kind of storage the accountants are using!



Example PDF invoice report generated with the Table to PDF node. The input data table is shown as image in the middle, and the custom title and author can be provided as flow variables.

The Results

Yes, they blend! We generated invoice reports with data from two different sources, Google Cloud Storage and Microsoft SharePoint. We defined the layout of the invoice report with the preprocessing nodes in KNIME, and let the workflow subsequently generate all invoice reports by one click, each one named according to the order id. A lot of manual work was saved. So, the next time if you receive your invoice report in record time, you might know the reason!

Google BigQuery Meets Databricks

Shall I Rent a Bike in this Weather?

Author: Emilio Silvestri, KNIME

Workflow on KNIME Community Hub: Google BigQuery Meets Databricks

The Challenge

“Life is like riding a bicycle. To keep your balance you must keep moving.” Despite its misuse under tons of Instagram pictures, this beautiful quote from Albert Einstein is still relevant today. In addition to physical activity, sustainable and shared mobility have become our weapon against daily traffic and pollution: Terms like shared transport, bike sharing, car sharing are now part of our language, and more people than ever use these services on a daily basis. How often are these services used? How is their usage affected by other factors, such as the quality of the service or the weather conditions?

To answer these questions, we need to collect data from a wide range of - typically disjointed - data sources, we also need a bit of imagination...and some patience! As an example, today we will mix together bike sharing data provided by Google BigQuery with weather data stored on Databricks, in order to see if and how weather conditions affect how the bikes are used.

For those who don’t know these two platforms, BigQuery is the Google response to the Big Data challenge. It is part of the Google Cloud Console and offers the ability to store and query large datasets using SQL-like syntax. Databricks is a cloud-based big data tool. Developed by the Apache Spark group, it offers a wide variety of operations - such as building data pipelines and scaling data science to production - tuning the functionalities offered by the Spark open-source software.

Both these platforms are supported by KNIME Analytics platform, from version 4.1 upwards. You can download and install the KNIME BigQuery and the KNIME Databricks Integration from the KNIME Community Hub.

Topic. Multivariate visualization of bike sharing data vs. weather data

Challenge. Investigate how weather influences usage of bike sharing

Access Mode / Integrated Tool. KNIME Google BigQuery Integration and KNIME Databricks Integration

The Experiment

The first dataset, hosted on Google BigQuery public data, is the Austin Bike Share Trips. It contains more than 600k bike trips during 2013-2019. For every ride it reports the timestamp, the duration, the station of departure and arrival, plus information about the subscriber. The second, smaller, dataset is the Austin Weather dataset, which is hosted on a Databricks platform. It contains daily weather information for the city of Austin, such as temperature, dew point, humidity, wind, precipitation, as well as adverse weather events.

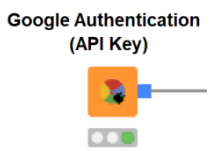
Google BigQuery

In the upper part of our KNIME workflow (which you can download from the KNIME Community Hub here) we access the Austin Bike Share Trips dataset hosted on the Google BigQuery platform as a public dataset. In order to execute this part of the workflow you need:

- An active project on Google Cloud Platform
- Credentials and an API key to access your project
- Google BigQuery JDBC driver installed on your KNIME Analytics Platform
- KNIME BigQuery extension and KNIME Google Cloud Storage Connection extension, both available on the KNIME Community Hub

Authentication

With the project credentials we are going to configure the Google Authentication (API Key) node. You will be required to provide your service account email and the P12 authentication file. You can find both on your Google Cloud Platform Project under: APIs & Services → Credentials

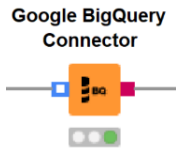


The Google Authentication (API Key) node.

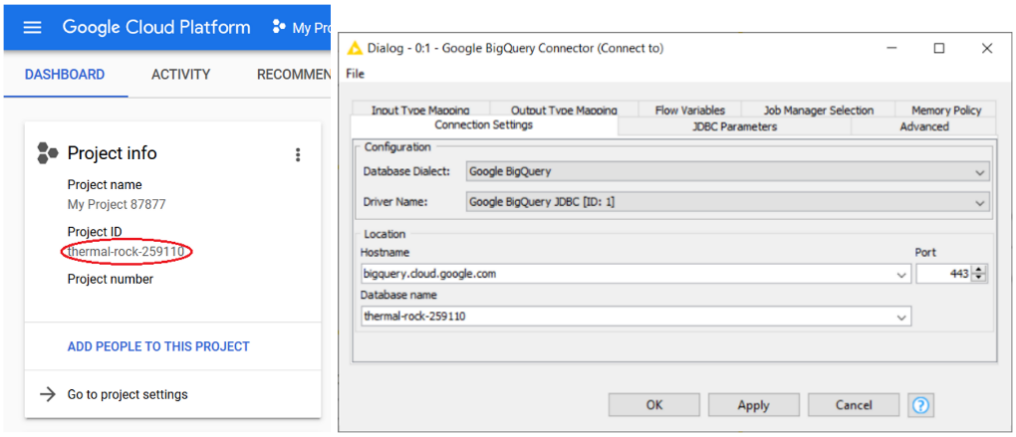
If you are starting from scratch with Google Cloud Platform, we recommend this step-by-step guide that will also show how to create a new project, generate new credentials and install the driver on KNIME Analytics Platform.

Connecting to Google BigQuery

After authentication, the Google BigQuery Connector node provides access to the public BigQuery platform, using the BigQuery JDBC Driver, the hostname (which is *bigquery.cloud.google.com*) and the Database name, that in this case is your Project ID. You can find the Project ID on your project's dashboard on the Google Cloud Platform.



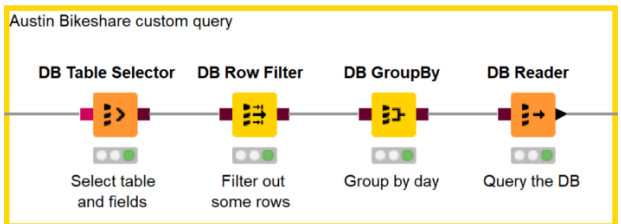
The Google BigQuery Connector node.



You can find your Project ID in the Google Cloud Platform. Add this ID in the "Database name" field in the configuration window of the Google BigQuery Connector node

Query

At this point, Google BigQuery has become your remote database and you can use all the DB nodes provided by KNIME Analytics Platform in the DB -> Query folder in the Node Repository panel. DB Query nodes are useful to build and execute powerful queries on the data before they are imported into your workflow. This is particularly



KNIME workflow section performing custom query on BigData.

useful when, as in this case, we are only interested in downloading a portion of the data and not the entire - *huge* - dataset.

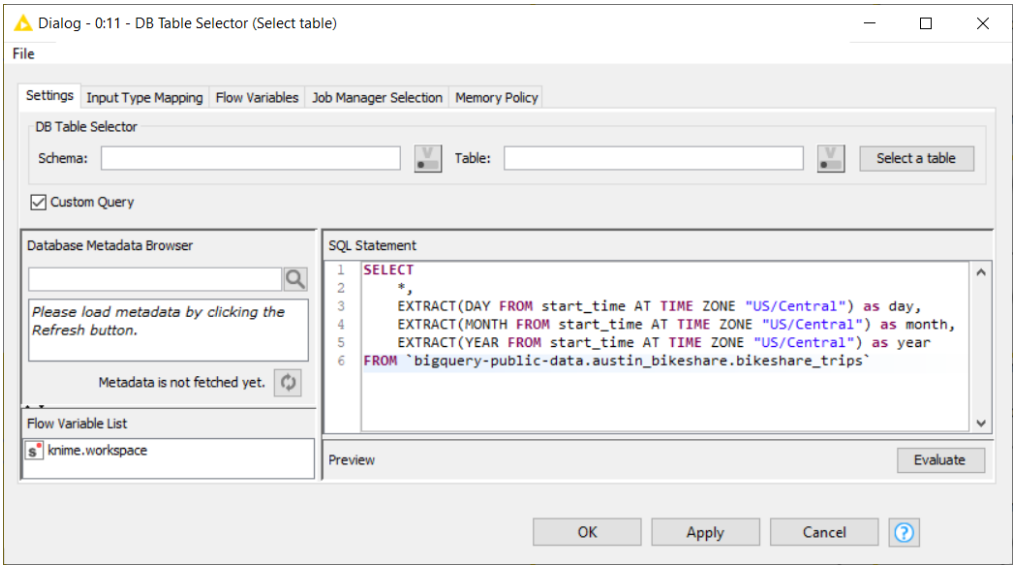
Let's add a DB Table Selector node and open the configuration window to write a custom query like the one shown in the configuration window below. It will extract some features such as year, month and year fields that we are using further down in the workflow.

> ***Note.*** When typing SQL statements directly, make sure to use the specific quotation marks (``) required by BigQuery.

We can refine our SQL statement by using a few additional GUI-driven DB nodes. In particular, we added a DB Row Filter to extract only the days in [2013, 2017] year range and a DB GroupBy node to produce the trip count for each day.

> ***Note.*** If you feel nostalgic about SQL queries, you can open the result window of every DB node (right click on the node -> last entry) and navigate to the "DB Query" tab to check how the SQL statement looks like so far.

Finally, we append the DB Reader node to import the data locally into the KNIME workflow.



DB Table Selector node configuration window with custom query

Databricks

The bottom part of the workflow handles the data stored on Databricks. What you need in this section is:

- A running Databricks cluster storing the Austin Weather dataset. You can download the csv file from Kaggle and upload it on a Databricks cluster.
- Credentials and cluster ID in order to connect to the Databricks instance
- The official JDBC driver provided by Databricks and installed on KNIME Analytics Platform (recommended)
- KNIME Databricks integration and KNIME Extension for Apache Spark available on the KNIME Community Hub

Note. KNIME Analytics Platform provides an open source Apache Hive driver that you can also use to connect to Databricks. However, it is strongly recommended to use the official JDBC driver provided by Databricks.

Connecting to Databricks

First of all, let's connect to Databricks adding the Create Databricks Environment node to the workflow. In the configuration window we are asked to provide a number of parameters:

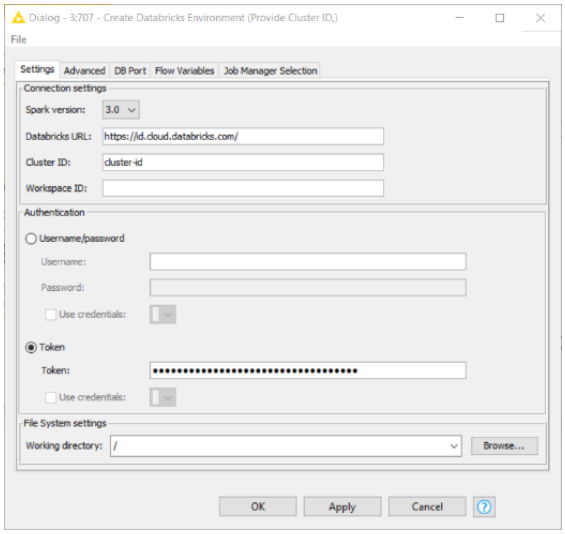
The Create Databricks Environment node.

1. Databricks URL
2. Cluster ID
3. Workspace ID
4. Authentication

Provide these parameters in the configuration window of the Create Databricks Environment node.

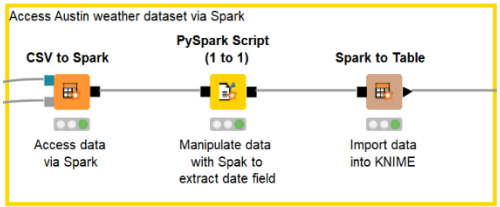
Executing this node connects KNIME Analytics Platform to the Databricks cluster where the data are stored. The node has three ports, each of them providing a different access to the data:

- Red port: JDBC connection to connect to KNIME database nodes.
- Green/blue port: DBFS connection to connect to remote file handling nodes as well as Spark nodes.
- Gray port: Spark context to connect to all Spark nodes. Please check in the Advanced tab of the configuration window that the option "Create Spark context" is enabled in order to activate this port.



The configuration window of the Create Databricks Environment node.

In this example we are going to use some basic Spark operations to retrieve the data. Please refer to the KNIME on DataBricks guide to explore further Databricks operations in KNIME Analytics Platform, plus more detailed instructions on how to configure Databricks for the first time.



KNIME workflow section manipulating and importing data from Databricks using Spark nodes

Query

Since we have stored the Austin Weather dataset on the Databricks cluster as a CSV file, let's add a CSV to Spark node to access it. Double click to open the configuration window. Click "Browse" and select the *austin_weather.csv* from the cluster.

At this point we are ready to use some of the functionalities offered by the Spark nodes. You can find them all in the Node Repository panel under Tools & Services -> Apache Spark, after installing the KNIME Extension for Apache Spark.

Here, we want to extract information regarding the date field. We are going to split the date string into three separate columns: year, month and day. To do so we use the PySpark Script (1 to 1) node and write our simple script directly into the configuration

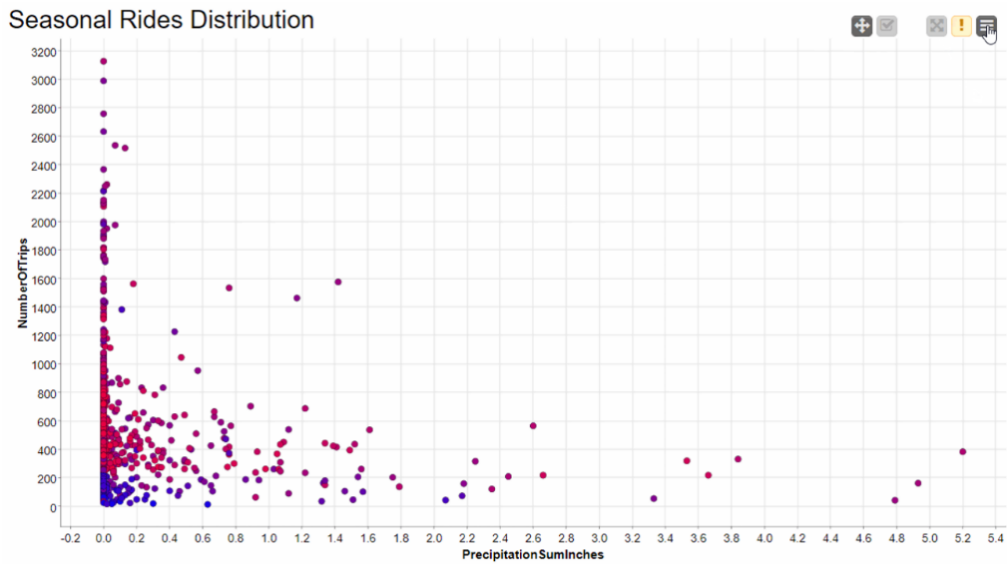
window. After execution, the output port contains the processed Spark data. Finally, a Spark to Table node imports the results into our KNIME workflow.

The Results

In the last steps we have extracted and prepared the data to be blended together.

Let's add a Joiner node, select the previously extracted *year, month and day* fields as joining columns and let KNIME Analytics Platform do its trick. After execution, right-click the Joiner node, select *Join result* from the menu and have a look at the data which are now blended.

Following the Joiner node in the workflow, the Visualization component builds a dashboard with different charts, such as bar chart, histogram, sunburst chart, and the table view and the scatter plot shown below. Each dot in the Scatter plot encodes the data of one day. The colour of the dot tells us about the average daily temperature: the colder days are blue, while the hotter are red.



Scatter plot of the blended data. Each dot encodes the number of bike rides for a specific day. It is colored according to the average daily temperature, blue for lower and red for higher values. We can explore different feature combinations directly from the Scatter plot interactive view.

The dashboard also offers some level of interactivity to dig into the exploration, such as an interactive slider to remove days according to the temperature level or a table showing only selected dots from the scatter plot. As shown above, we can also change the configuration of the chart directly from the dashboard, choosing different feature combinations clicking the list icon in the upper right corner.

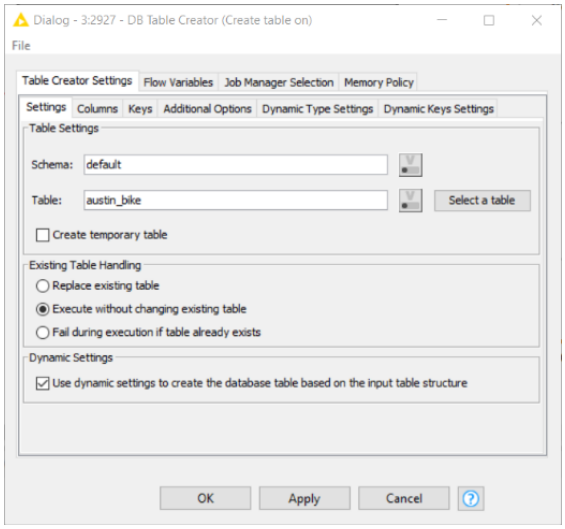
For example, we can get information about the relation between bike rides and rain level, choosing the corresponding features - *PrecipitationSumInches* for the X axis and *NumberOfTrips* for the Y. From the resulting scatter plot, we can see that during the days with the higher numbers of rides there was hardly any rain or no rain at all: the bad weather conditions might have led people to choose different means of transportation.

Let's now click the list icon again and select *Date* column for the X axis. The scatter plot updates revealing a seasonal trend of the bike rides. We can explore the details of the data points with the higher number of bike rides by selecting them from the scatter plot and having a look at the Table view. It seems as if the peaks we can see mostly take place during March and October - when biking is probably more pleasant than during rain or at very high or very low temperatures.

At this point, we might want to upload the blended data back to the platforms, for future uses. In order to do so, let's add a DB Table Creator to the workflow. We can connect it either to the Google BigQuery Connector or to the DB connection (Red port) of the Create Databricks Environment node.

Note that additional steps such as the creation of a new schema in your personal BigQuery project might be necessary.

Configure the DB Table Creator node selecting the desired schema and giving a name to the table, then append and configure a DB Loader node to upload the data to the new remote table.



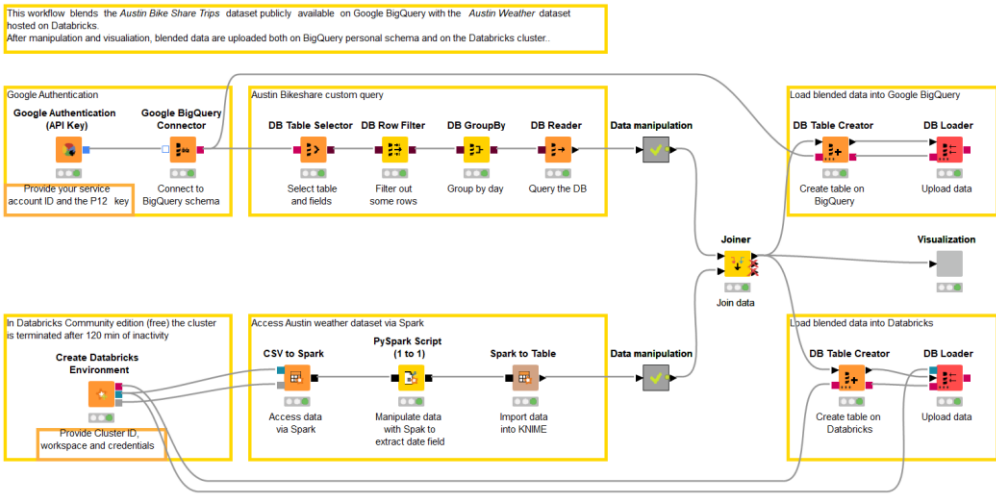
DB Table Creator configured to create a new table named austin_bike in the default schema in Databricks.

Note. When using BigQuery, remember to delete all the space characters from the column names. They would be automatically renamed during table creation, and this will create conflict for the next step, since column names will no longer match.

Wrapping up

In this example we have learned how to access and blend two popular cloud services - Google BigQuery and Databricks - using the extensions available in KNIME Analytics Platform. Together with these datasets, we have explored the world of bike sharing and how its usage is intrinsically related to weather conditions.

The full Google BigQuery Meets Databricks workflow for this experiment is shown below and is available for download from the KNIME Community Hub.



The final Google BigQuery Meets Databricks workflow blending data from BigQuery and Databricks, available for download from the KNIME Community Hub.

Amazon ML Services Meets Google Charts

Travel Risk Guide for Corporate Safety

Authors: Jim Falgout, Maarit Widmann, Lada Rudnitckaia, KNIME

Workflow on KNIME Community Hub: Amazon ML Services Meets Google Charts

The Challenge

Due to the increasing internationalization of business, more and more employees are becoming globally mobile. Providing reliable information and protecting workers abroad is their employer's duty. This means, among other things, assessing risks and implementing risk management. Therefore, before you start planning a business trip, for example, to Germany, let's first take a look at how safe it is there!

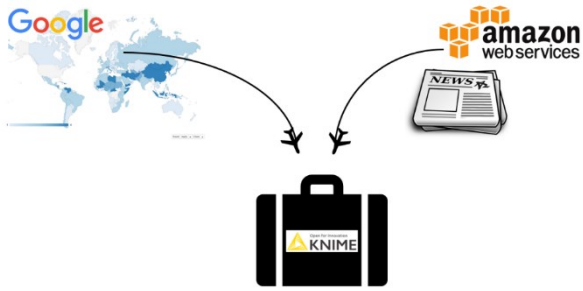
Our travel safety guide consists of two parts:

In the first part, we put together an interactive travel risk map to show the risk level in all countries as estimated by the U.S. Department of State. The map is a choropleth world map based on Google Charts and JQuery library, easy to build with this Choropleth World Map component. The travel advisory information is retrieved via the RSS feed of the U.S. Department of State - Bureau of Consular Affairs web page.

In the second part, we put together an interactive dashboard that shows the general atmosphere in a country by

1. a tag cloud with the key phrases in the news and
2. a table with news extracts and their sentiments

We use the KNIME integration with Amazon ML Services to translate the news, determine their sentiments, and extract the key phrases from them.



Our travel risk guide visualizes the risk levels in each country in a choropleth map based on Google Charts and shows information extracted from local news and analyzed with Amazon ML Services.

Topic. Show travel risk information in a choropleth world map. Translate, extract key phrases, and determine the sentiments of the local news in one country.

Challenge. Visualize numeric information assigned to country names in a choropleth map based on Google Charts and JQuery library. Perform natural language processing tasks using Amazon ML Services integrated in KNIME Analytics Platform.

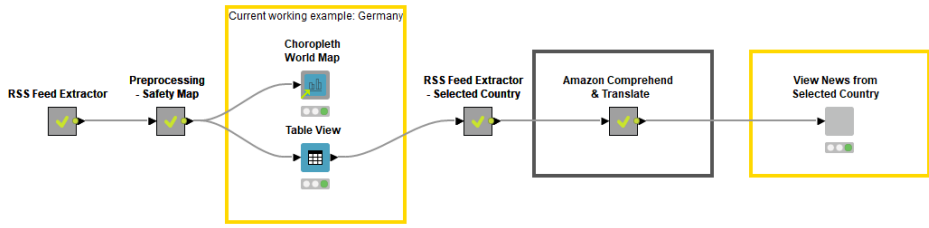
Access Mode / Integrated Tool. RSS feed, Google Charts, Amazon ML Services

The Experiment

Visualizing travel risks in a choropleth map based on Google Charts

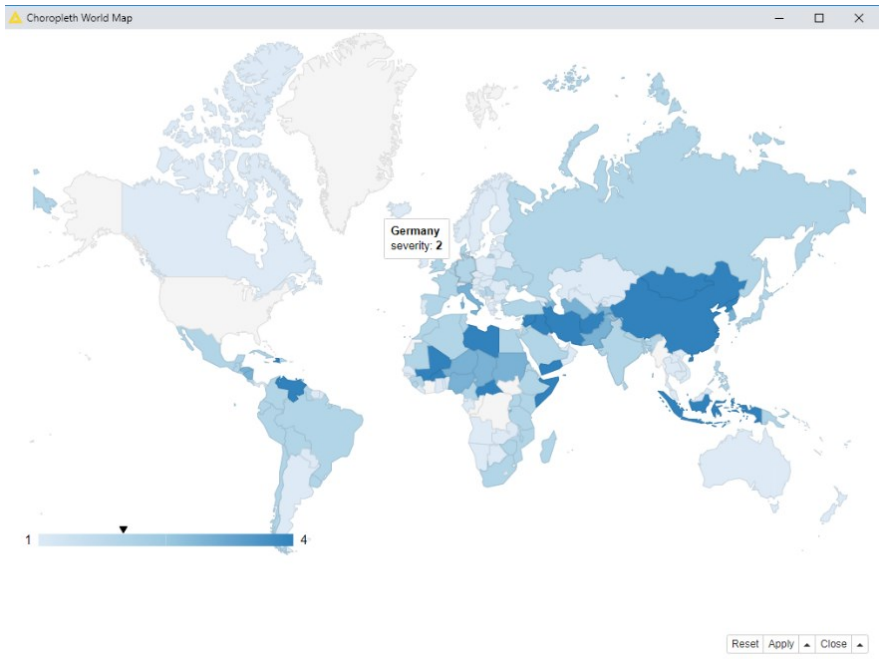
The choropleth map shows the travel risk levels for each country as estimated by the U.S. Department of State. The travel advisories are provided as short sentences, for example, "Iceland - Level 1: Exercise Normal Precautions". We extract the numeric risk levels (from level 1 "Exercise Normal Precautions" to level 4 “Do Not Travel”) and country names into separate columns and use these columns to build the Choropleth world map. These steps make the first part of the workflow shown below. The Travel Risk Map for Corporate Safety using AWS Comprehend and Translate workflow is available on the KNIME Community Hub.

1. Inside the “RSS Feed Extractor” metanode we access the travel advisories of the U.S. Department of State with the RSS Feed Reader node. The information for each country is provided in a separate document.

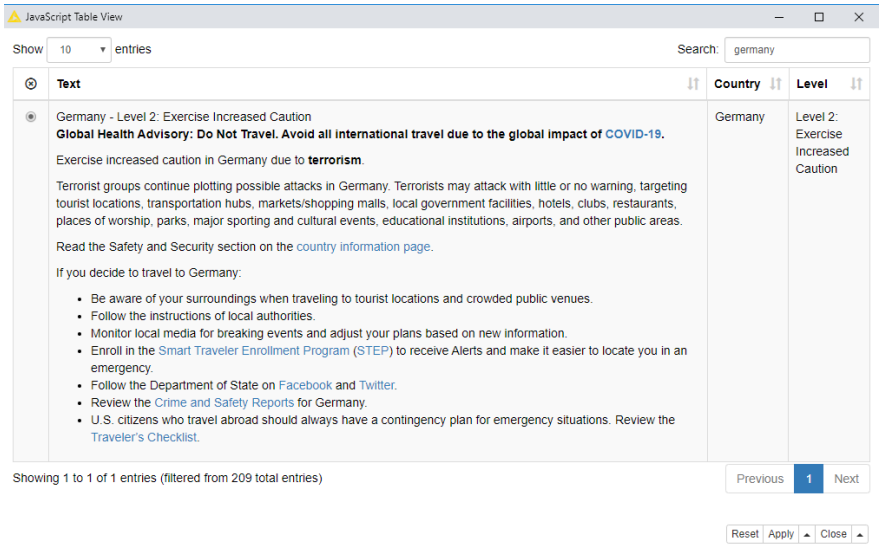


Workflow that 1) visualizes travel risk levels in countries in a choropleth world map, and 2) extracts and translates keywords and sentiments of local news in a specific country. The choropleth map is based on Google Charts, the natural language processing tasks are performed using Amazon ML Services, both integrated in KNIME Analytics Platform as components/nodes with a graphical dialog. You can download the Travel Risk Map workflow on the KNIME Community Hub

2. Inside the "Preprocessing - Safety Map" metanode, we separate the country names and risk levels from the documents and manipulate some country names so that they match the country names that are used to build the choropleth map.
3. Next, we use the Choropleth World Map component to visualize the risk level in each country.
4. As a last step in the first part of our travel safety guide, we use the Table View node that displays travel advisory information about each country in an interactive table and allows for selecting one or multiple countries for further analysis. We select the country of our interest, in our case Germany.



A choropleth world map visualizing the travel risk level (the darker the riskier) in each country as estimated by the U.S. Department of State.



Interactive table displaying travel advisory information for each country, as estimated by the U.S. Department of State. One or multiple countries can be selected in the view for further analysis, where the sentiments and key phrases of local news are investigated.

Translating, extracting keywords, and analyzing the sentiment of local news with Amazon ML Services

Now that we have obtained the global picture, we want to analyze Germany in more detail. What are people talking about? Are they optimistic, or rather scared? For convenience, we will also not try to translate German texts ourselves but let Amazon ML Services translate the local news for us. These steps are covered in the second half of the workflow.

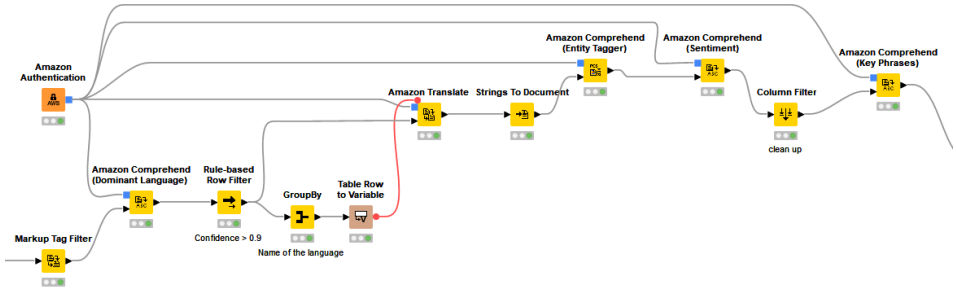
We start by retrieving some German news via the RSS feed of two providers, ntv and t-online. Next, we use Amazon ML Services to determine the dominant language of the retrieved documents, translate them into English, and determine their sentiment positive/negative/mixed/neutral. Finally, we extract the key phrases from the documents. All these steps can be performed with nodes that are part of the KNIME Amazon Machine Learning Integration, all with graphical user interfaces.

Amazon Web Services and KNIME Amazon Machine Learning Integration

Amazon Web Services (AWS) cloud platform provides services for several machine learning (ML) and artificial intelligence (AI) capabilities. KNIME Analytics Platform integrates various AWS services, and here we use two of them: Comprehend for ML

based natural language processing and Translate for neural machine translation of text from a source to a target language. You can install the KNIME Amazon Machine Learning Integration from the KNIME Community Hub.

1. **Connecting to AWS:** Amazon ML services is a paid service, and all Amazon Web Services nodes require an AWS connection. Therefore, we start with the Amazon Authentication node. We provide the authentication with, for example, an access key ID and secret key. We can get these credentials if we log in to the AWS Management Console, and select "My Security Credentials" in the user menu in the top right corner. In the page that opens, we click "Create access key", and make sure to save the secret key, which can't be retrieved afterwards. In the configuration dialog of the Amazon Authentication node, we check the "Access Key ID and Secret Key" option and enter the credentials in the fields that activate. Once the authentication has been provided, we can check "Switch Role", and enter the account ID and the desired role that has been granted some specific permissions.
2. **Using nodes in the KNIME Amazon Machine Learning integration:** Next, we perform a number of natural language processing tasks. The key nodes here are:
    a. Amazon Comprehend (Dominant Language): takes a document column as input and shows its dominant language in a separate column in the output
    b. Amazon Translate: takes the original document in German (or another source language) as input and outputs the document translated into English (or another target language) in a string column
    c. Amazon Comprehend (Sentiment): takes the translated document as input, and assigns a sentiment into each document, available in a separate column in the node's output
    d. Amazon Comprehend (Key Phrases): extracts the key phrase (a noun and its modifiers) from the input document into a string column



Connecting to AWS and using Amazon ML services for natural language processing tasks, such as determining the dominant language, analyzing the sentiment, and extracting the key phrases of documents, via their dedicated nodes in KNIME Analytics Platform.

If you're interested in how to integrate other Amazon Machine Learning capabilities in KNIME Analytics Platform, check out this KNIME and AWS Machine Learning Integration blog post.

Visualizing key phrases in a tag cloud and displaying news extracts with their sentiments

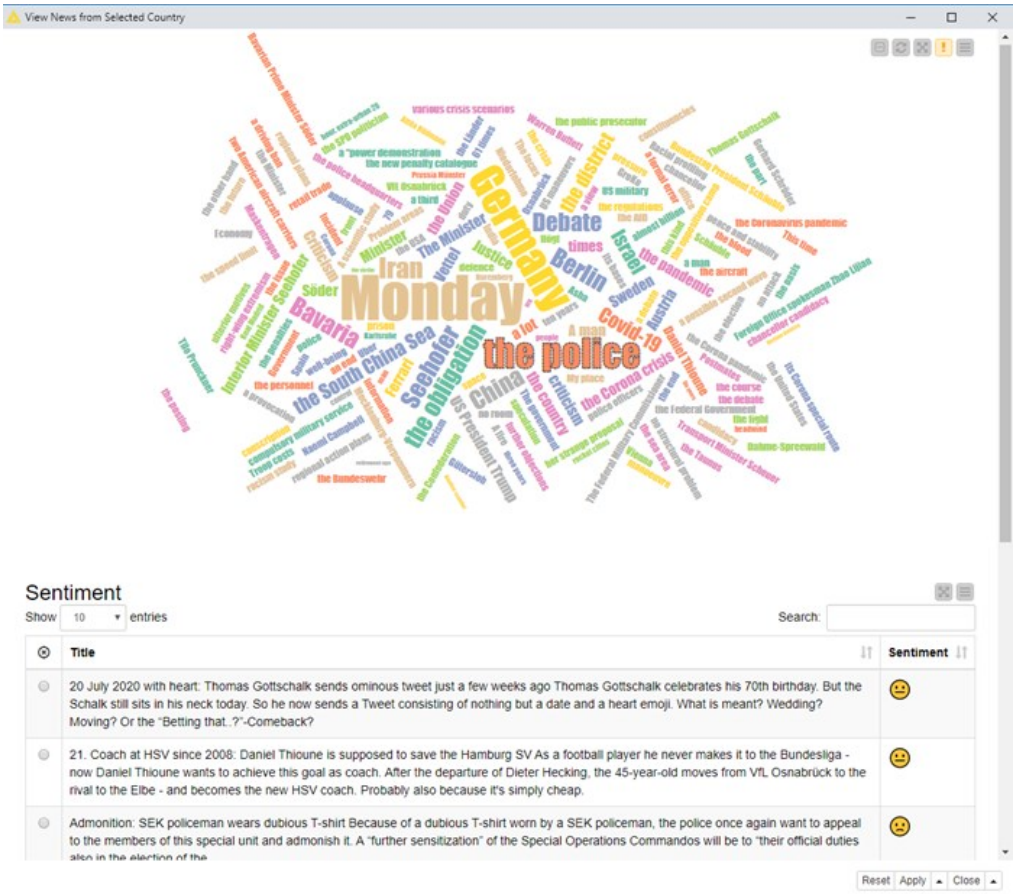
In the last step, we visualize the extracted information in another interactive dashboard. Inside the "View News from Selected Country" component, we calculate the numbers of occurrences of the different key phrases and use the count of each phrase to define its size in the tag cloud. We also show a list of translated German news and their sentiments.

The result is shown in the interactive view output of the component.

The Results

Yes, they blend! The final workflow successfully blends a global overview of travel risks, shown on a Google choropleth map, and news alerts for a selected country, translated, key phrases extracted, and sentiments determined using Amazon ML Services.

So - can we travel to Germany? According to the U.S. Department of State, we should be careful at public places, and exercise increased caution, as advised by risk level 2. It seems that Germany is talking about covid-19, and about some common topics in international and national politics, as given by, for example, the key phrases "the minister" and "the pandemic".



Tag cloud showing the most common topics in German news, and a list of translated news and their sentiments, based on the RSS feed of ntv and t-online. The news were translated, and their sentiments and key phrases were determined using the KNIME Amazon Machine Learning Integration.

BIRT Meets Tableau & JavaScript

How was the Restaurant?

Authors: Scott Fincher, KNIME

Workflow on KNIME Community Hub: BIRT and Tableau: Will they Blend?

The Challenge

How would your favorite restaurant perform in a health inspection? Are you sure you are eating in one of the most reliable and safest restaurants in town? If you live in Austin, Texas, we can check.

Over the last three calendar years, data.austintexas.gov has published a dataset of restaurant inspection scores for Austin (TX) restaurants. Inspection scores range between 0 (theoretically) and 100. An inspection score lower than 70 requires corrective measures; repeated low scores may even necessitate the closure of the restaurant.

We will use this dataset to visually explore:

- How many restaurants have been inspected in each ZIP code area;
- Of those restaurants how many have scored between 70 and 80, 80 and 90, and 90 and 100, as well as perfect 100 scores;
- And finally, the average scores for ZIP code locations in the Austin area.

For each one of these goals, we will use a:

- pie chart;
- grouped bar chart;
- geolocation map.

So far so good. Now we need to choose the graphical tool for such representations.

Since data manipulation within a reporting environment might be clunky, many KNIME users prefer to run the data manipulation comfortably from KNIME Analytics Platform, and later export the data to their preferred reporting tool.

There are many ways to produce graphics in a report with KNIME Analytics Platform. Today we will consider three: with native graphics in BIRT, with JavaScript based images exported to BIRT, with native graphics in Tableau.

BIRT (Business Intelligence Reporting Tool) is a reporting tool, which to a certain extent, is distributed as an open-source tool. The KNIME Report Designer extension integrates BIRT within KNIME Analytics Platform. The "Open Report" button in the KNIME tool bar takes you to the BIRT report environment. The "KNIME" button in the BIRT report environment takes you back to the KNIME workbench.

Tableau Desktop is a reporting tool, which requires a commercial license. 14-day trial licenses are also available at the Tableau Desktop site. The KNIME Tableau extension allows you to communicate with Tableau Desktop via .hyper files or directly with the Tableau Server.

Note. Starting from version 4.2 of KNIME Analytics Platform Tableau integration uses the "Tableau Hyper API" which uses the .hyper format. If you want to use the .tde format for a Tableau Server 10.4 or earlier you can install and setup the legacy extension as described in the Legacy Extensions section.

It is also possible to produce the graphic image in the KNIME workflow via the JavaScript nodes and subsequently export the image to the reporting tool of your choice.

In today's Will They Blend, we want to highlight the process of generating a few simple graphics using BIRT, JavaScript, and Tableau. Can we produce three simple charts using those tools? What might that involve? Let's find out!

Topic. Data Visualization using BIRT, Tableau, and JavaScript

Challenge. Create a Pie Chart, a Bar Chart, and a geo-location map using BIRT, Tableau, and JavaScript

Access Mode. KNIME Tableau extension, Report Designer extension, JavaScript Views extension, Open Street Map Integration extension

The Experiment

For this particular experiment, we need the following KNIME extensions:

- Tableau extension,

- Report Designer extension,
- JavaScript Views extension,
- Open Street Map Integration extension

We also need a licensed installation of Tableau Desktop.

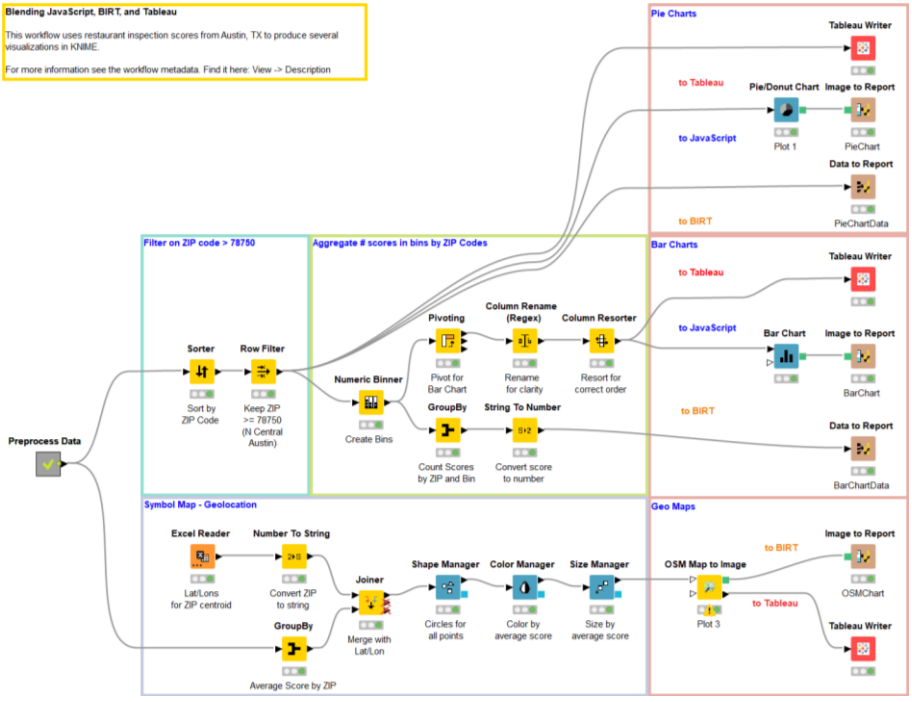
We first pre-process the data in KNIME to obtain only the most recent inspection score for each restaurant.

Next, we produce three different plots, in order of complexity:

- A simple pie chart of number of inspections by ZIP Code in North Central Austin
- A grouped bar chart showing binned restaurant scores by ZIP code in North Central Austin
- A symbol map of average scores by ZIP code for the entire Austin area

In the following, we will discuss how to prepare the data for visualization using data manipulation nodes inside a KNIME workflow. Then we will describe how the data are exported to build the plots in BIRT, Tableau, and JavaScript.

The workflow used is shown below and is available on the KNIME Community Hub: [BIRT and Tableau: Will they Blend?]



The workflow for this blend processes restaurant inspection scores to produce three different charts, in BIRT, Tableau, and JavaScript. The [BIRT and Tableau: Will they Blend?] workflow is available on the KNIME Community Hub.

KNIME for Data Preparation

As shown above, the pre-processed data are passed out of a metanode along a few different paths.

- The first path, shown in the area outlined in aqua above, is for restricting analyses to a subset of ZIP codes in the North Central Austin area.
- For the grouped bar chart, additional processing was performed using the KNIME nodes in the area outlined in green. First, bins for the score data were created using the Numeric Binner node. Inside this path, two sub-paths are started to count the scores for each bin:
    - with a GroupBy node to prepare data for a BIRT bar chart
    - with a Pivoting node to prepare data for a JavaScript based bar chart and for a bar chart in Tableau.
- Finally, the last branch of the workflow creates the symbol map using the nodes shown in the area of the KNIME workflow outlined in light blue. First, average scores for each ZIP code were calculated using a GroupBy node. A table of latitude and longitude values for the centroid of each ZIP were read in using the Excel Reader node, and the ZIPs were converted to strings with a Number to String node.

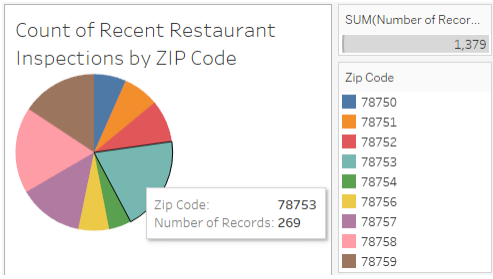
We are ready to plot.

Pie Charts

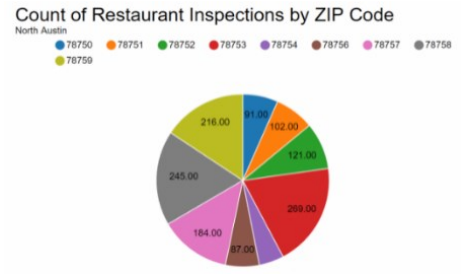
The upper right light red square contains 3 simple strategies to plot the input data to a pie chart. We want the pie chart to show the number of restaurants per ZIP code. Thus, the input data are produced by the first Row Filter node.

- **To Tableau**. The node named "Tableau Writer" generates a Tableau .hyper file of the input data. Double clicking the exported .hyper file brings up the Tableau interface, with our data preloaded. Inside the Tableau platform, "ZIP code" was selected as the *dimension*, and "SUM(Number of Records)" as the *measure*. In the "Show Me" tab, we selected *Pie Chart*, and in the "Marks" shelf, we assigned "ZIP code" to "*Color*" and "SUM(Number of Records)" to "*Size*" and "*Angle*".
- **To JavaScript**. Here the Pie/Donut Chart node plots the pie chart of number of restaurants inspections by ZIP code. This node, like all JavaScript View nodes, produces an interactive view and a static image, assuming this has been set in its configuration window. In order to produce a report, the image of the pie chart is sent to an Image to Report node to be exported into the BIRT reporting environment.

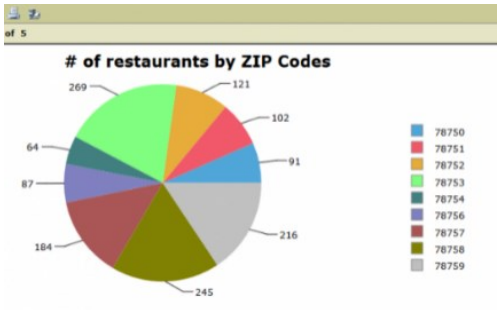
- **To BIRT**. The Data to Report node exports the input data into the BIRT reporting environment. Here, a chart is built using "ZIP code" as row category, with "Grouping" enabled and function "Count", and using "Score" as slice size definition in the "Select Data" tab.



Pie Chart with Tableau.



Pie Chart with JavaScript.



Pie Chart with BIRT.

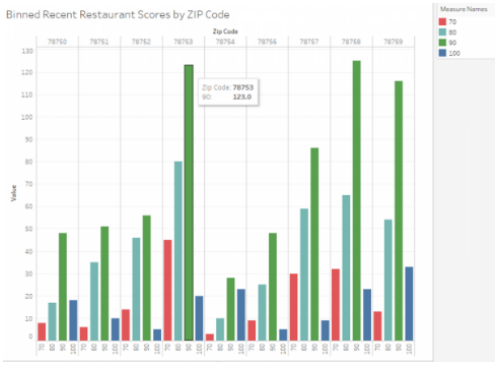
Bar Charts

A similar approach was adopted to generate the bar charts.

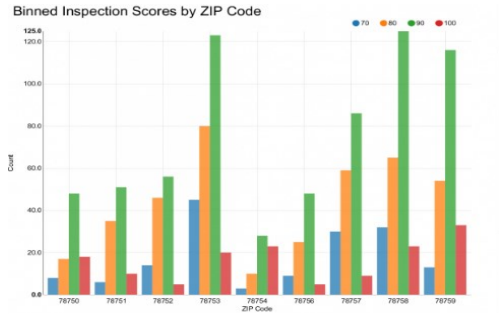
- The Tableau Writer node to export the bar chart data into a .hyper Tableau formatted file. Inside Tableau, we selected SUM(70), SUM(80), SUM(90), and SUM(100) as our measures and placed them on the *Rows* Similarly, we used "ZIP code" as a dimension and placed it on the *Columns* shelf. We selected *Side-by-side Bars* from the "Show Me" tab. We applied "Measure Names" to *Color* in the *Marks* shelf.
- The Bar Chart node to plot the number of scores in each bin into a bar chart image, and an Image to Report node to include the image into a BIRT report.
- The Data to Report node to export the bar chart data into the BIRT reporting environment. Here, a chart is built using "ZIP code" as "Category (X) Series" with

Grouping enabled and function “Sum” and “Count(Score)” as “Category (Y) Series” in the “Select Data” tab.

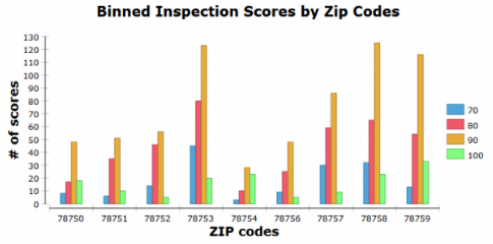
Notice that while the Pivoting node is preparing the data for the bar chart in Tableau and JavaScript, the GroupBy node produces the bar chart data in the right format to be used by BIRT.



Bar Chart with Tableau.



Bar Chart with JavaScript.



Bar Chart with BIRT.

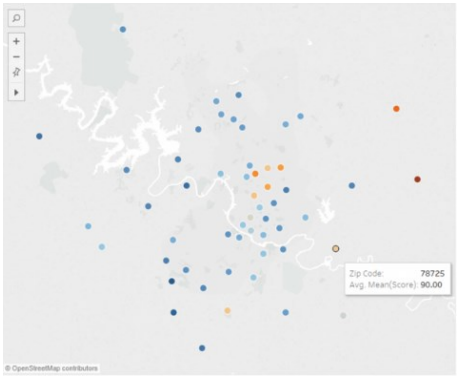
Geo-location Map

For the geo-location map, we calculated the average score for each ZIP code area with a GroupBy node. After combining the averages scores with the lat/lons using a Joiner node, a series of Manager nodes set the shape, color, and size of the ZIP code center points.

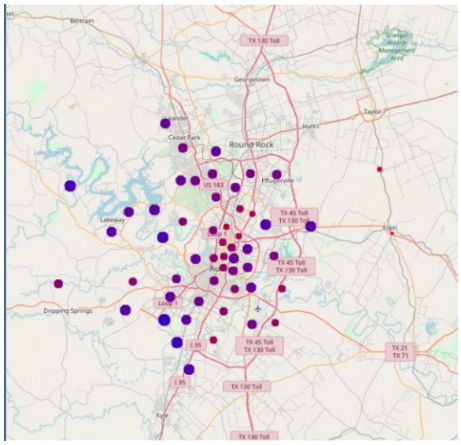
Finally, a symbol map is generated using the “OSM Map to Image” node, and passed to BIRT with an “Image to Report” node as in the previous steps.

The same ZIP code center points are passed to the Tableau Writer node. The .hyper file produced by this node is then imported into Tableau. We selected *Symbol Map* from the “Show me” tab. In the “Marks” shelf, with a mark type *Circle*, with AVG(Score) to *Color* and ZIP Code to *Detail*. We also changed the color palette to “Orange-Blue Diverging” in order to see distinctions among the scores a bit better.

Notice that for this plot, there is no purely BIRT based solution, since the open source BIRT version does not offer a geo-location map. The remaining two solutions - with Tableau and with a JavaScript generated image exported to BIRT – are shown in the figures below.



Geo-map with Tableau.



Geo-map with JavaScript.

The Results

As you can see, we set out to build pie charts presenting the number of restaurant inspections for each ZIP code.

We also produced a grouped bar chart showing binned restaurant scores by ZIP code.

Finally, we produced a geolocation map, showing average scores for each ZIP code according to their location coordinates.

The data were processed, blended, joined, cleaned, and colored using KNIME native nodes. The pie chart, bar chart, and geo map were generated with JavaScript, BIRT, and Tableau.

Thus, in conclusion, we can say once again “yes - they blend!”

Of course, there are many more visualization options available in each platform. Experiment on your own and see what you can come up with!

Twitter Meets Azure

Sentiment Analysis via API

Authors: Craig Cullum, Data Scientist

Workflow on KNIME Community Hub: Sentiment with Azure

The Challenge

Staying on top of your social media can be a daunting task, Twitter and Facebook are becoming the primary ways of interacting with your customers. Social media channels have become key customer service channels, but how do you keep track of every Tweet, post, and mention? How do you make sure you're jumping on the most critical issues, the customers with the biggest problems?

As Twitter has become one of the world's preferred social media tools for communicating with businesses, companies are desperate to monitor mentions and messages to be able to address those that are negative. One way we can automate this process is through Machine Learning (ML), performing sentiment analysis on each Tweet to help us prioritize the most important ones. However, building and training these models can be time consuming and difficult.

There's been an explosion in all of the big players (Microsoft, Google, Amazon) offering Machine Learning as a Service or ML via an Application Programming Interface (API). This rapidly speeds up deployment, offering the ability to perform image recognition, sentiment analysis and translation without having to train a single model or choose which Machine Learning library to use!

As great as all these APIs can be, they all have one thing in common. They require you to crack open an IDE and write code, create an application in Python, Java or some other language.

What if you don't have the time? What if you want to integrate these tools into your current workflows? The REST nodes in KNIME Analytics Platform let us deploy a workflow and integrate with these services in a single node.

In this 'Will They Blend' article, we explore combining Twitter with Microsoft Azure's Cognitive Services, specifically their Text Analytics API to perform sentiment analysis on recent Tweets.

Topic. Use Microsoft Azure's Cognitive Services with Twitter.

Challenge. Combine Twitter and Azure Cognitive Services to perform sentiment analysis on our recent Tweets. Rank the most negative Tweets and provide an interactive table for our Social Media and PR team to interact with.

Access Mode / Integrated Tool. Twitter & Microsoft Azure Cognitive Services.

The Experiment

As we're leveraging external services for this experiment we will need:

- A Twitter developer account
- An Azure account

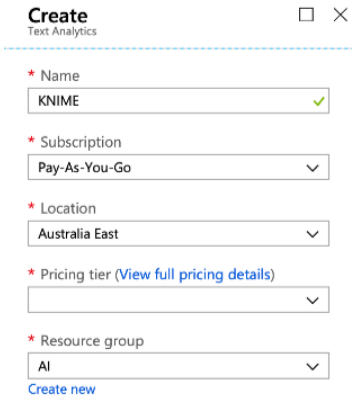
You'll need your Twitter developer account's API key, secret, Access token and Access token secret to use in the Twitter API Connector node. You'll also want your Azure Cognitive Services subscription key.

Creating your Azure Cognitive Services account

When you log in to your Azure Portal. Navigate to Cognitive Services and we'll create a new service for KNIME.

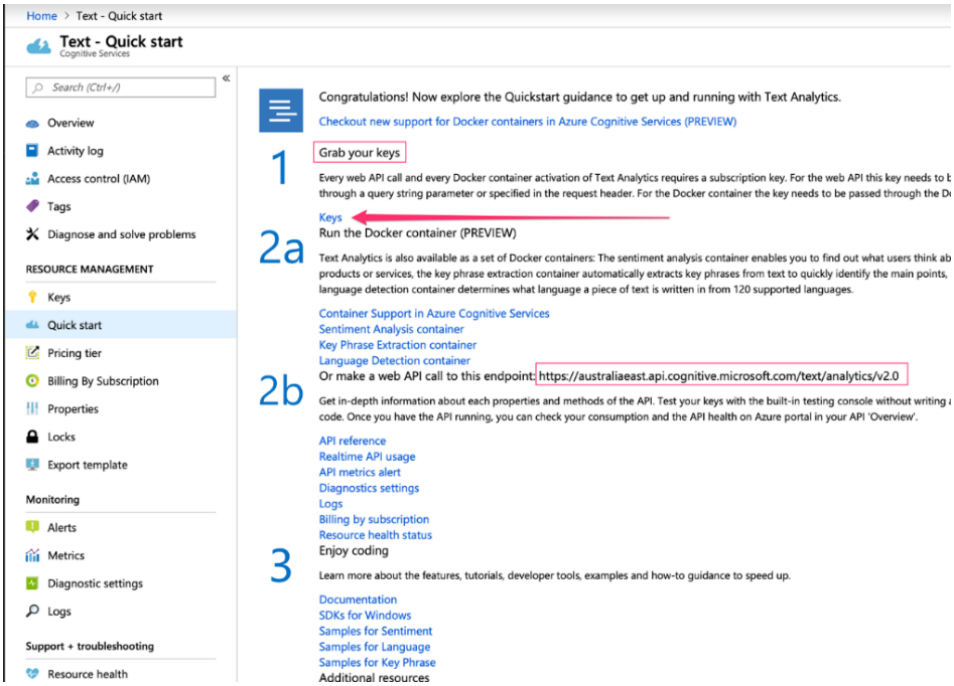
1. Click add and search for the Text Analytics service.

2. Click Create to provision your service giving it a name, location and resource group. You may need to create a new Resource group if this is your first Azure service.



Click Create to provision your service giving it a name, location, and resource group.

3. Once created, navigate to the Quick Start section under Resource Management where you can find your web API key and API endpoint. Save these as you'll need them in your workflow.



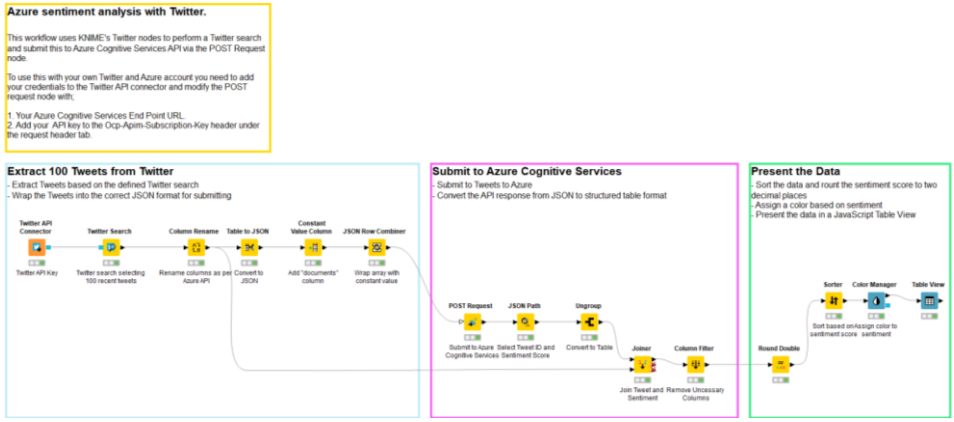
The Quick Start section in Resource Management where you can find your web API key and API endpoint

Extracting Tweets and passing them to Azure Cognitive Services

Deploying this workflow is incredibly easy, in fact, it can be done in just 15 nodes.

The workflow contains three parts that take care of these tasks:

1. Extracting the data from Twitter and wrapping them into a JSON format that is compatible with the Cognitive Services API
2. Submitting that request to Cognitive Services
3. Taking the output JSON format and turning it into a structured table for reporting. Ranking the sentiment and applying a color



Workflow using Twitter nodes to perform a Twitter search and submit this to Azure Cognitive Services API via the POST request node.

Azure expects the following JSON format:

```
{
  "documents": [
    {
      "id": "1",
      "text": "I loved the meal"
    },
    {
      "id": "2",
      "text": "I left hungry and unhappy"
    },
    {
      "id": "3",
      "text": "The service was excellent"
    }
  ]
}
```

KNIME Analytics Platform includes excellent Twitter nodes that are available from KNIME Extensions if you don't already have them installed. This will allow you to

quickly and easily connect to Twitter and download tweets based on your search terms.

We can take the output from Twitter, turn it into a JSON request in the above format and submit. The Constant Value Column node and the JSON Row Combiner node wrap the Twitter output with the document element as expected.

The POST Request node makes it incredibly easy to interact with REST API services, providing the ability to easily submit POST requests.

You'll need to grab the respective URL for your region, here in Australia the URL is;

```
https://australiaeast.api.cognitive.microsoft.com/text/analytics/v2.0/sentiment
```

We can leave the Authentication blank as we'll be adding a couple of Request Headers.

We need to add for the Header Key;

```
Content-Type
```

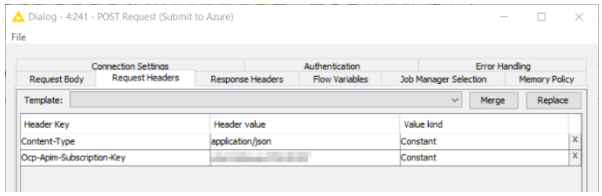
and Header Value;

```
application/json
```

And another Header Key;

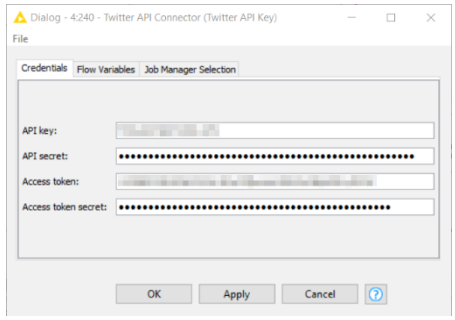
```
Ocp-Apim-Subscription-Key
```

The Header Value for the Subscription Key will be the key provided as part of your Azure Cognitive Services you created.



The Header Value for the Subscription Key is the key provided as part of the Azure Cognitive Services you created

If you're using the KNIME Workflow as a guide, make sure to update the Twitter API Connector node with your Twitter API Key, API Secret, Access token and Access token secret.



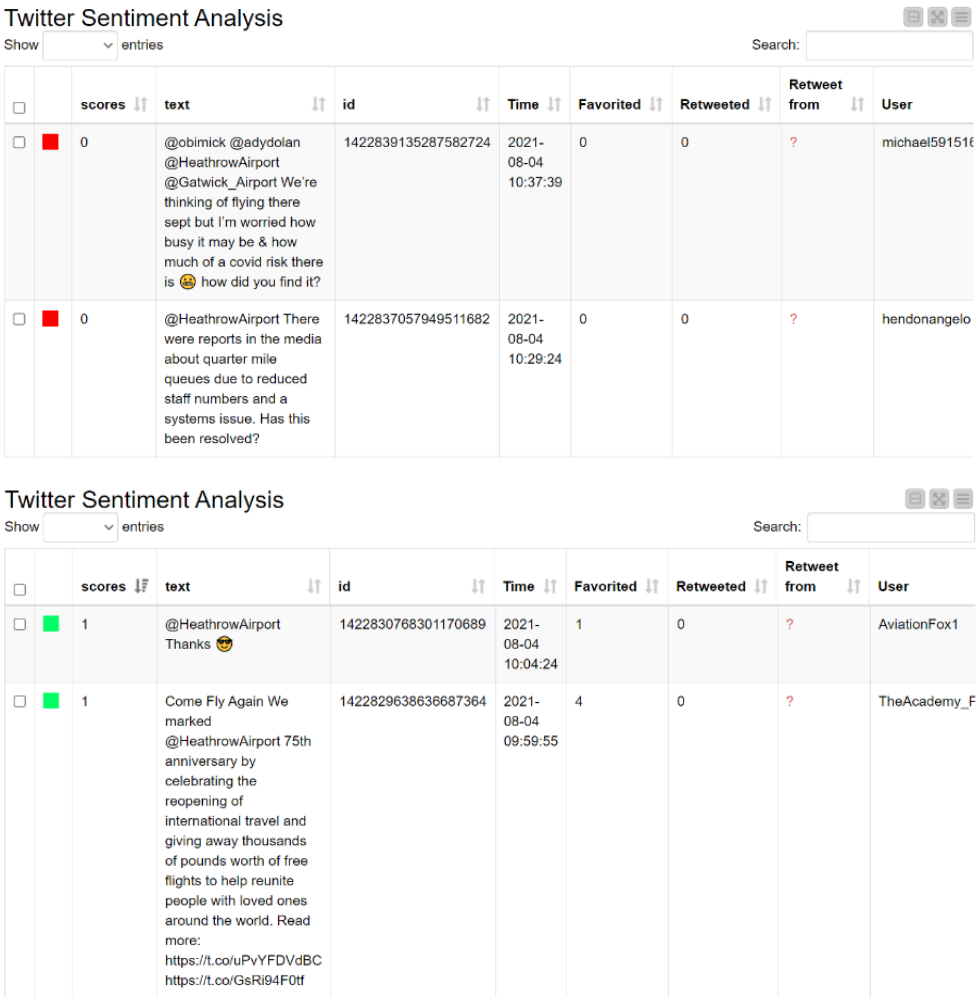
Update the Twitter API Connector node with your Twitter API Key, API Secret, Access token and Access token secret.

We can now take the response from Azure, ungroup it, and join these data with additional Twitter data such as username and number of followers to understand how influential this person is. The more influential they are, the more they may become a priority.

Reporting the Data

Once created you can use a Table View node to display the information in an interactive table, sorted by sentiment. This can be distributed to PR and Social Media teams for action, improving customer service.

To really supercharge your KNIME deployment and make this service truly accessible, you can use the KNIME Business Hub to create an interactive online sentiment data app for your social media team allowing them to refresh reports, submit their own Twitter queries or provide alerting so your team can jump on issues.



Twitter Sentiment Analysis

Show entries Search:

	scores	text	id	Time	Favorited	Retweeted	Retweet from	User
	0	@obimick @adydolan @HeathrowAirport @Gatwick_Airport We're thinking of flying there sept but I'm worried how busy it may be & how much of a covid risk there is 😬 how did you find it?	1422839135287582724	2021-08-04 10:37:39	0	0	?	michael59151
	0	@HeathrowAirport There were reports in the media about quarter mile queues due to reduced staff numbers and a systems issue. Has this been resolved?	1422837057949511682	2021-08-04 10:29:24	0	0	?	hendonangelo

Twitter Sentiment Analysis

Show entries Search:

	scores	text	id	Time	Favorited	Retweeted	Retweet from	User
	1	@HeathrowAirport Thanks 😎	1422830768301170689	2021-08-04 10:04:24	1	0	?	AviationFox1
	1	Come Fly Again We marked @HeathrowAirport 75th anniversary by celebrating the reopening of international travel and giving away thousands of pounds worth of free flights to help reunite people with loved ones around the world. Read more: https://t.co/uPvYFDVdBC https://t.co/GsRi94F0tf	1422829638636687364	2021-08-04 09:59:55	4	0	?	TheAcademy_F

Interactive table in the data app on KNIME Business Hub visualizing results.

So, were we able to complete the challenge and merge Twitter and Azure in a single KNIME workflow? Yes, we were!

References:

- You'll find this 15-node workflow on the KNIME Community Hub here
- Twitter Data on the KNIME Community Hub

Snowflake Meets Tableau

Data Warehouse in an Hour?

Authors: Craig Cullum, Data Scientist

Workflow on KNIME Community Hub: Snowflake Meets Tableau

The Challenge

It's Friday morning, your boss comes to your desk and has grand data plans! They've promised the Board of Directors a flashy new dashboard to report sales figures and you're the person to do the job. Oh, and they need it by the end of the day!

Traditionally, this would have been a mammoth undertaking. Carefully planning your data structure, spending months gathering requirements and purchasing hardware.

Nothing kills any data project quicker than if:

- It doesn't contain the data they need
- End users find it slow
- New data source can't be added easily
- Data are incorrect and dirty

It is important to plan our future state to prevent that from happening. Luckily, the in-cloud data and analytics space is moving quickly, vendors are coming out with new, easy to use, easy to integrate solutions all the time. Rather than building our database from scratch, custom coding our ETL processes or fighting with archaic report writing solutions we go Agile and straight to the web and find a Software as a Service solution that fits the bill.

One application that is getting a huge amount of traction in the marketing place is Snowflake. Snowflake is a Software as a Service (SaaS) cloud data platform that can be deployed on Azure, AWS, or GCP globally. Snowflake takes a fresh approach with scaling, both performance and cost. Automatically switching off (and therefore the cost) when users aren't using the database or automatically scaling to respond to a sudden increase in user load ensuring performance.

By connecting KNIME Analytics Platform to Snowflake, we can quickly create and populate an in-cloud data warehouse that's enterprise ready, high performing and works with the market leading reporting and business intelligence tools.

As many organizations have already established BI tools, such as Tableau, we will use Tableau as the reporting tool that will connect to our Snowflake data. Tableau Desktop is a reporting tool that requires a commercial license.

Trials are available for both Tableau and Snowflake. KNIME Analytics Platform supports Tableau with its KNIME Tableau Integration extension that allows you to save data to Tableau's in-memory format .Hyper and also publish data directly to Tableau Server.

With the flexibility of KNIME, you can choose whatever in-cloud data warehouse or Business Intelligence package that best aligns with your business. We will describe a DWH workflow, populating a Snowflake instance daily with sales data from Excel and filtering out all returned products provided to us from our online e-commerce platform in JSON format, downloaded via a REST API. A subset of the daily data is then exported to a *.hyper* Tableau file to produce the final report.

Will Snowflake and Tableau blend?

Topic. Create an in-cloud Data Warehouse (DWH).

Challenge. Create a DWH workflow to import sales data to Snowflake, blend this data with return information from our e-commerce platform in JSON format from the REST API, and produce a report in Tableau. Will they blend?

Access Mode / Integrated Tool. Snowflake & Tableau

The Experiment

For this experiment we will need:

- A trial or licensed installation of Tableau.
- A trial or licensed Snowflake instance.

This Snowflake meets Tableau workflow is available on the KNIME Community Hub.

Preparing the Data before Uploading to Snowflake

We have 10,000 orders provided to us from the sales team in Excel format. We will be using the Excel file Sample – APAC Superstore.xls provided by Tableau as a sample data source. You can find this within the My Tableau Repository/Datasources folder on your local computer after you've downloaded and installed Tableau Desktop.

The issue we have with this data is that they also include returned orders, so we will need to filter returned orders out of the dataset if we are to get an accurate figure for Sales and Profit.

We will use the Excel Reader node to read the Excel file and perform some simple transformations to filter out Customer and Product IDs from the dataset, convert a string to Date&Time and round our values to two decimal places.

Our returns data is imported in JSON format from our online e-commerce platform's REST API, we will read this data into KNIME, convert it to a tabular format and blend this data with our order data to get an accurate dataset.

Joining the Datasets

We use the JSON Path node to output Order ID and Returned columns and then use Ungroup to return the dataset to a traditional looking data table for joining to the Excel data, converting semi-structured to structured data.

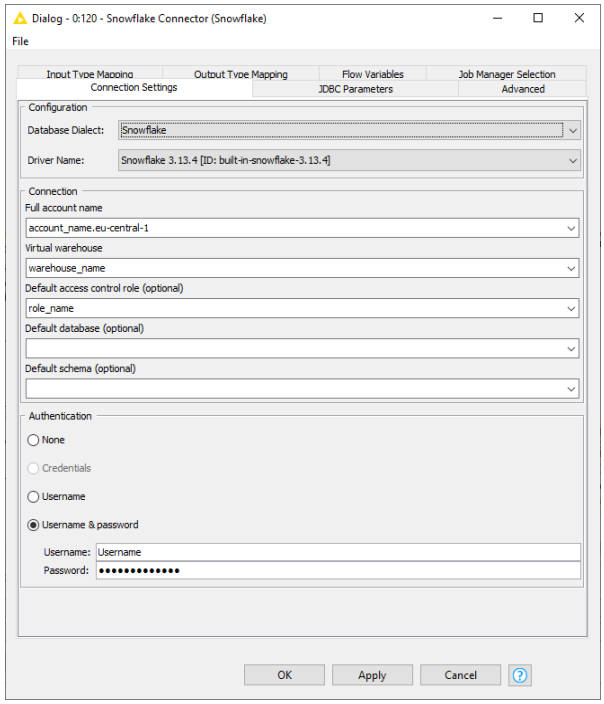
Let's join the two datasets together with a Joiner node. Because not all orders have a value for Returned we want to select Left Outer Join as the Join mode and our Top Input and Bottom Input should both be set to Order ID.

We then leverage a Row Filter node and exclude rows by attribute value, testing the Returned column and matching using pattern matching "Yes".

Uploading the Data to Snowflake

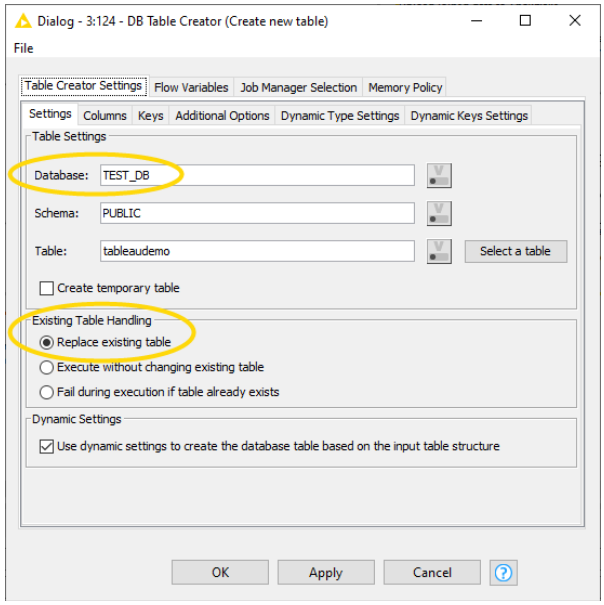
Dynamically creating tables based on input data is extremely powerful within KNIME Analytics Platform. This removes the headache of having to constantly edit SQL code to create tables as your dataset changes and new columns are added.

Starting from version 4.4, KNIME Analytics Platform includes a dedicated Snowflake Connector node with the built in JDBC driver. In the configuration of this node, all you need to provide is the full account name, virtual warehouse, and the credentials. Note that an existing warehouse is needed already at this step.



Configuring the Snowflake Connector node: providing the full account name, virtual warehouse, and credentials.

Once connected to Snowflake, the standard KNIME Database extension nodes will allow selecting not only the schema and the table but also the database.



The configuration window of the DB Table Creator node. The Standard KNIME Database extension nodes connected to Snowflake allow to additionally select the database.

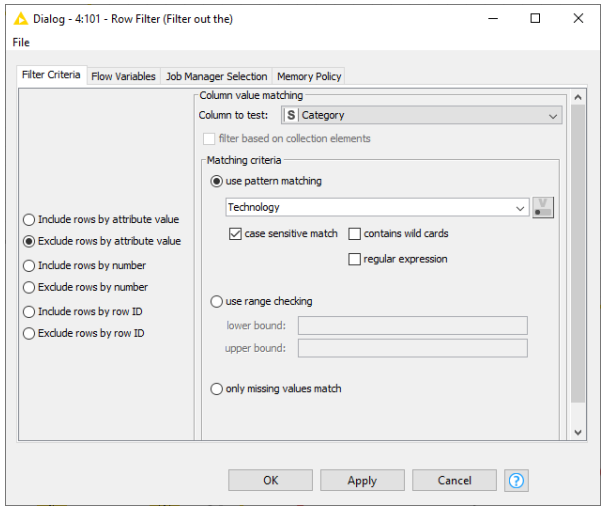
Add a DB Table Creator, to create a new database table and to drop any pre-existing table that may have been created. To prevent any failures during your workflow, configure your DB Table Creator node and check “Replace existing table”. Enter the database, schema, and the table name, ensure our SQL Types are correct. You want to make sure the “Order Date” and “Ship Date” fields are set to date.

The DB Loader node takes input from our Column Filter node to upload the data to a Snowflake stage and load the data into an existing table via the copy command.

Once you run these two nodes, your table will be created and the data uploaded to your cloud data warehouse ready for Tableau reporting.

Connecting the exchanged DB output port of the DB Loader node with the DB Reader node, I can bring this data back into KNIME for validation. As I’m not interested in the Technology division, I can use a Row Filter node to filter the dataset to only include the Furniture and Office Supplies division prior to exporting to Tableau in Tableau’s .Hyper format to distribute amongst my executive team. I can write the data into the Hyper file using the Tableau Writer node.

Note. Starting with version 4.2 of KNIME Analytics Platform, the Tableau integration builds on the new Tableau Hyper API which uses the .hyper format and ensures better compatibility and effortless installation. If you want to use the .tde format for a Tableau Server 10.4 or earlier you can install and setup the legacy extension.



Using a Row Filter node to filter the dataset to exclude the Technology division, i.e. only include the Furniture and Office Supplies division prior to exporting to Tableau in Tableau’s .hyper format.

The Results

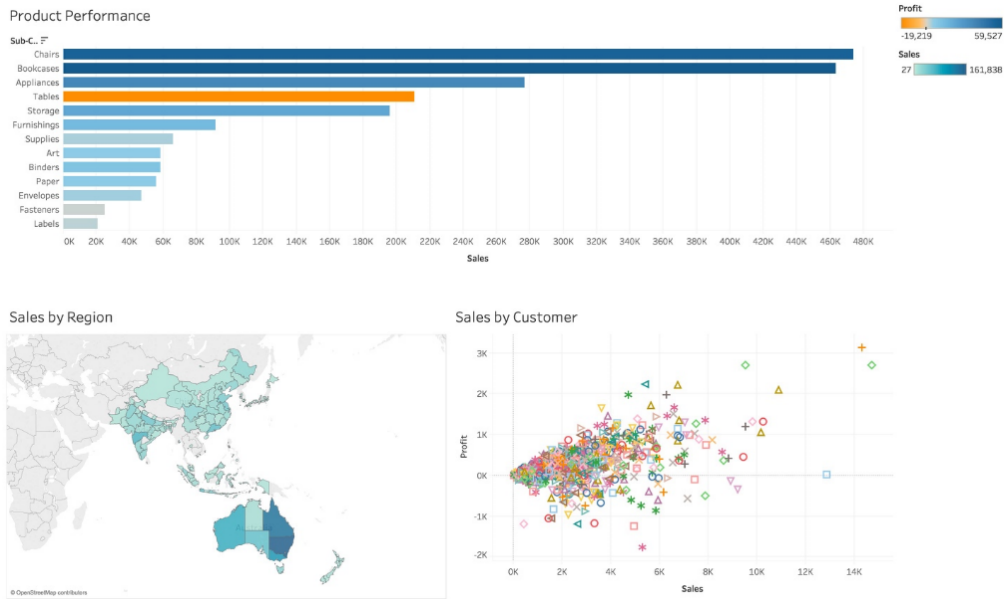
I can open Tableau Desktop and open the data directly via the exported .Hyper file containing my filtered clear data ready for my presentation.

As my requirements change and the use of data matures in my organization, I can expand my KNIME workflows to include additional datasets, calculations or predictive modelling.

The results show that although I have a healthy amount of sales of Tables, by color coding each bar with profit, from orange to blue, we can see that we are in fact losing money on tables.

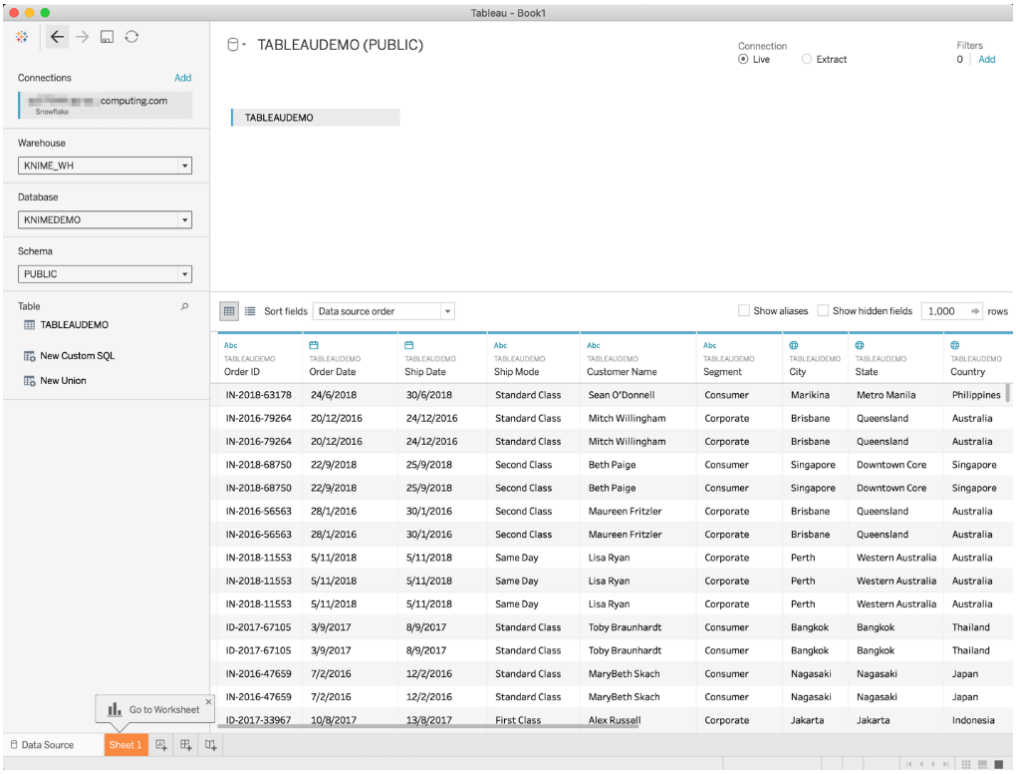
The map highlights the concentration of sales geographically, assisting the organization with decisions around warehousing, shipping and other logistics. We may choose for example to create a warehouse on the east coast of Australia as our sales are concentrated in this region.

Finally, our scatter plot shows customers plotted against profit and sales, this visualization allows an organization to pinpoint our most and least valuable customers.



Opening the Tableau Desktop, the data from the exported .hyper file contains precisely the data required for presentation – Product Performance, Sales by Region, and Sales by Customer.

Armed with this dashboard, I can share this amongst the senior leadership team, not only showing current sales but, by providing additional context, improve our insight and decision-making capabilities.



Excerpt of the Sales data, provided by Tableau as a sample data source.

KNIME Meets Semantic Web

Ontologies – or let's see if we can serve pizza via the semantic web and KNIME Analytics Platform

Authors: Martyna Pawletta, KNIME

Workflow on KNIME Community Hub: Exploring A Pizza Ontology with an OWL file

Ontologies study concepts that directly relate to "being" i.e., concepts that relate to existence and reality as well as the basic categories of being and their relations. In information science, an ontology is a formal description of knowledge as a set of concepts within a domain. In an ontology we have to specify the different objects and classes and the relations - or links - between them. Ultimately, an ontology is a reusable knowledge representation that can be shared.

Fun reference. The Linked Open Data Cloud has an amazing graphic showing how many ontologies (linked data) there are available in the web.

The Challenge

The Semantic Web and the collection of related Semantic Web technologies like RDF (Resource Description Framework), OWL (Web Ontology Language) or SPARQL (SPARQL Protocol and RDF Query Language) offer a bunch of tools where linked data can be queried, shared and reused across applications and communities. A key role in this area is played by ontologies and OWLs.

So where does the OWL come into this? Well, no - we don't mean the owl as a bird here – but you see the need of ontologies, right? An OWL can have different meanings, and this is one of the reasons why creating ontologies for specific domains might make sense.

Ontologies can be very domain specific and not everybody is an expert in every domain - but it's a relatively safe bet to say that we've all eaten pizzas at some point in time - so let's call ourselves pizza experts. Today's challenge is to extract information from an OWL file containing information about pizza and traditional pizza toppings, store

this information in a local SPARQL Endpoint, and execute SPARQL queries to extract some yummy pizza, em - I mean data. Finally, this data will be displayed in an interactive view which allows you to investigate the content.

Topic. KNIME meets the Semantic Web.

Challenge. Extract information from a Web Ontology Language (OWL) file.

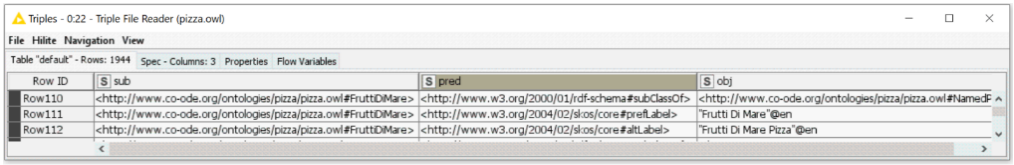
Access Mode / Integrated Tool. KNIME Semantic Web/Linked Data Extension.

The ontology used in this blog post and demonstrated workflow is an example ontology that has been used in different versions of the Pizza Tutorial run by Manchester University. See more information on Github here.

The Experiment

Reading and querying an OWL file

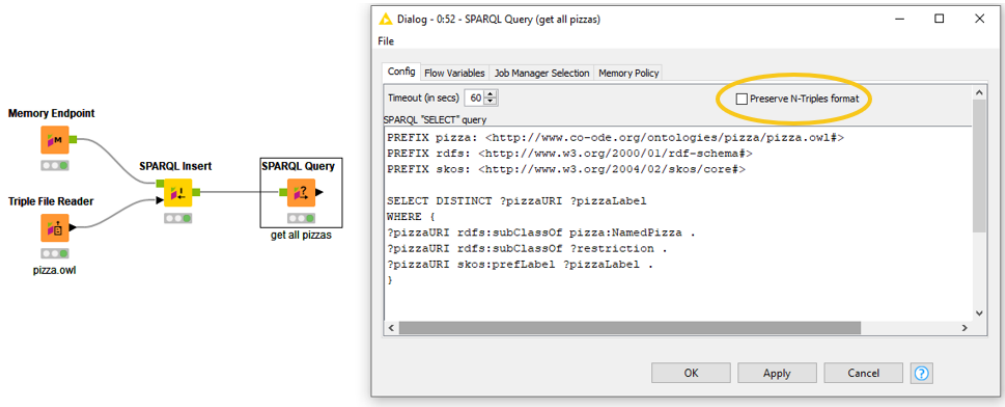
In the first step the Triple File Reader node extracts the content of the pizza ontology in the OWL file format and reads all triples into a Data Table. Triples are a collection of three columns containing a subject(URI), a predicate(URI) and an object(URI or literal), short: sub, pred, obj. The predicate denotes relationships between the subject and the object. As shown in the screenshot below, in the example we see that the Pizza *FruttiDiMare* is a *subClassOf* the class *NamedPizza* and has two labels: a preferred and an alternative one.



Screenshot showing the output of the Triple File Reader node containing a subject, predicate and object column.

Once the Triple File Reader is executed, a SPARQL Endpoint can be created using the Memory Endpoint together with the SPARQL Insert node. This allows the execution of SPARQL queries. Note that our Triple File Reader does not officially support the OWL format. KNIME can read RDF files and consequently because OWL files are very similar we can read these files too. However not all information is necessarily retrieved as OWL can have additional parameters.

The example below shows a SPARQL query node that contains a query to extract a basic list with all pizzas included in the owl file.

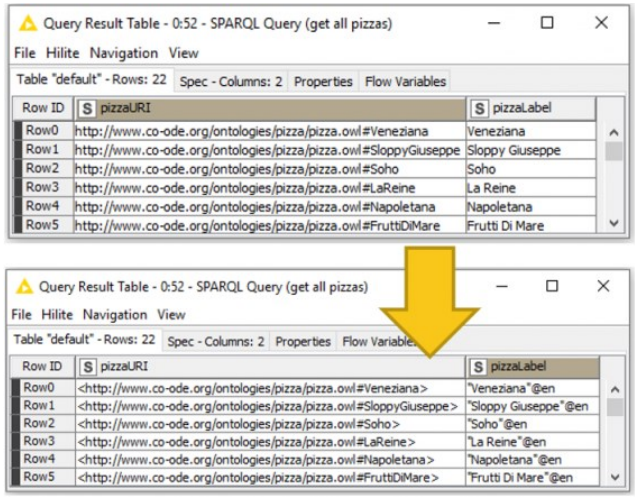


Example workflow that shows how to read an OWL file, insert extracted triples into a SPARQL endpoint and execute a SPARQL query to extract all kinds of pizzas from the pizza ontology.

A recommendation here: if the ontology you want to query is new to you – I would highly recommend exploring the structure and classes first quickly in another tool like Protége. This makes it easier later to create and write SPARQL queries.

The SPARQL query node has a checkbox on the top right saying “Preserve N-Triples format”. Selecting this makes a difference in terms of what the output data will look like. The N-Triples format needs to be kept if the triples will be inserted into an endstore.

The example below shows the effect of not checking (top) or checking (bottom) the N-triples checkbox. In case of URIs the angled brackets are not preserved, in terms of literals quotes and type (here @en) will be removed if nothing has been selected.

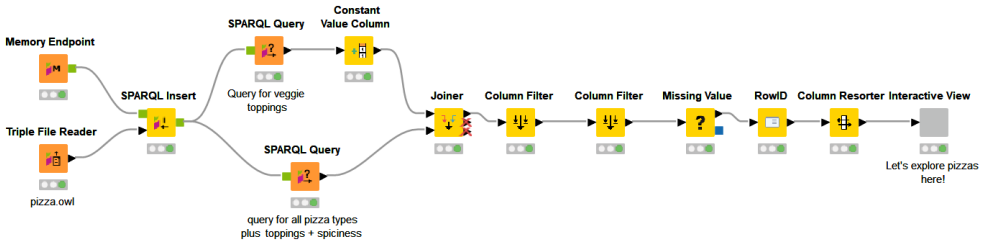


The effect of not checking (top) or checking (bottom) the N-Triples checkbox.

Visualization

There are different ways in KNIME to visualize data. In the case of ontologies, it's really depending on what you are aiming to do. Here we will extract a bit more information than in the first example and create an interactive view within a component that allows us to explore the content of the pizza ontology.

Additionally to the pizza labels, we now using two SPARQL query nodes with which further information like toppings per pizza type or its spiciness was extracted. Also, we query for pizza toppings that are a subclass of the class *VegetableToppings* and create a flag if the topping is a vegetable or not using the Constant Value Column node.



Example workflow showing how the basic example from above can be extended and an interactive view created.

Finally, we create an interactive view (see below) where the extracted data can be explored. To open the interactive view, right click the "Interactive View" Component + select Interactive View.

Is it real?!

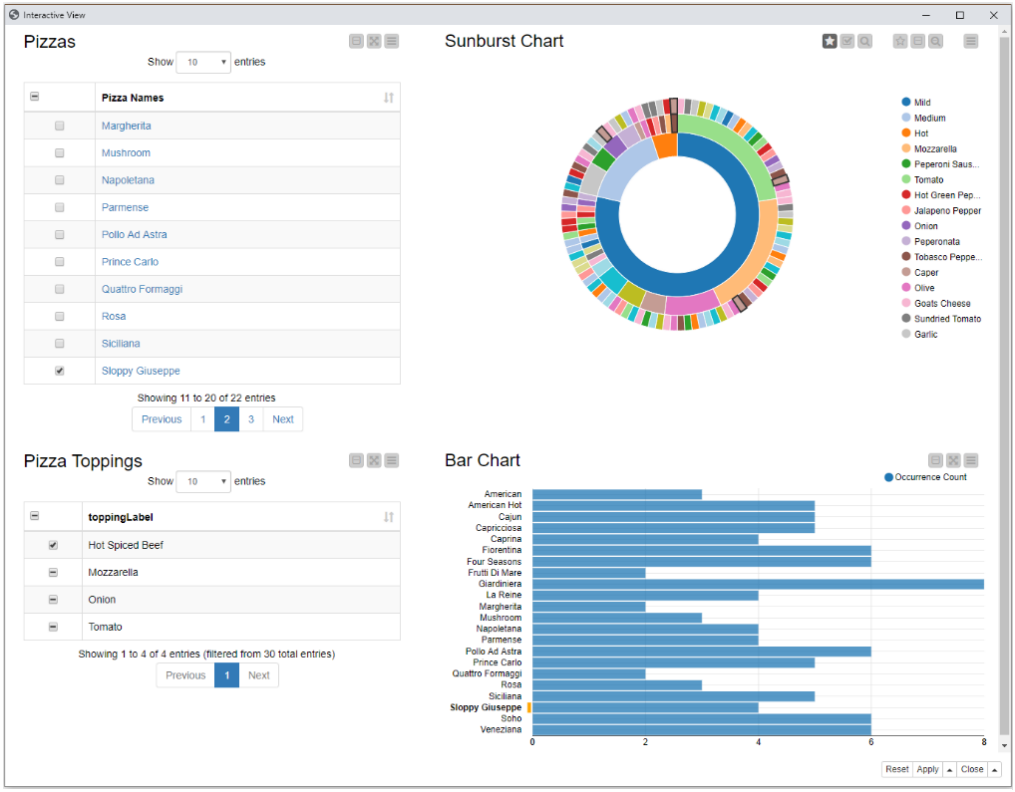
When I first looked at the dataset using the view, I saw the "**Sloppy Giuseppe**" Pizza and directly had to google it as it was something completely new to me. I saw the toppings but was wondering if this is something really Italian? This brought me to the idea of adding another feature here in addition to the tables and charts.

If you now click on the pizza name, a new window will open showing the Google search results for that specific pizza type. I did this using the String Manipulation node, which creates a link. To make sure the link opens in a new window and not in your current view the "target=_blank" option needs to be included.

Expression

```
1 string("<a href='https://google.com/search?q="+$pizzaLabel$+"+Pizza'"+"target=\"_blank\">"+$pizzaLabel$+"</a>")
```

The expression inside the String Manipulation node. For each pizza name, this creates a link that opens the Google search result in a new window.



Interactive view showing extracted data.

The Results

We showed today how to extract data from an OWL file, create a SPARQL Endpoint and SPARQL query. Finally, we generated a view where the content can be explored.

After playing with such yummy data… hungry now? Let's order a pizza then 😉

The example workflow shown in this article, Exploring a Pizza Ontology with an OWL file, can be downloaded from the KNIME Community Hub.

OCR on Xerox Copies Meets Semantic Web

Have Evolutionary Theories Changed?

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

Workflow on KNIME Community Hub: Will they blend? OCR meets Semantic Web - Natural Selection vs Modern Theory

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 are 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. Before installing this extension (see how to install KNIME Extensions) you will have to enable the Experimental software site (see 2.2.1. Enabling the Experimental software site). Let's continue step by step, after the package installation.

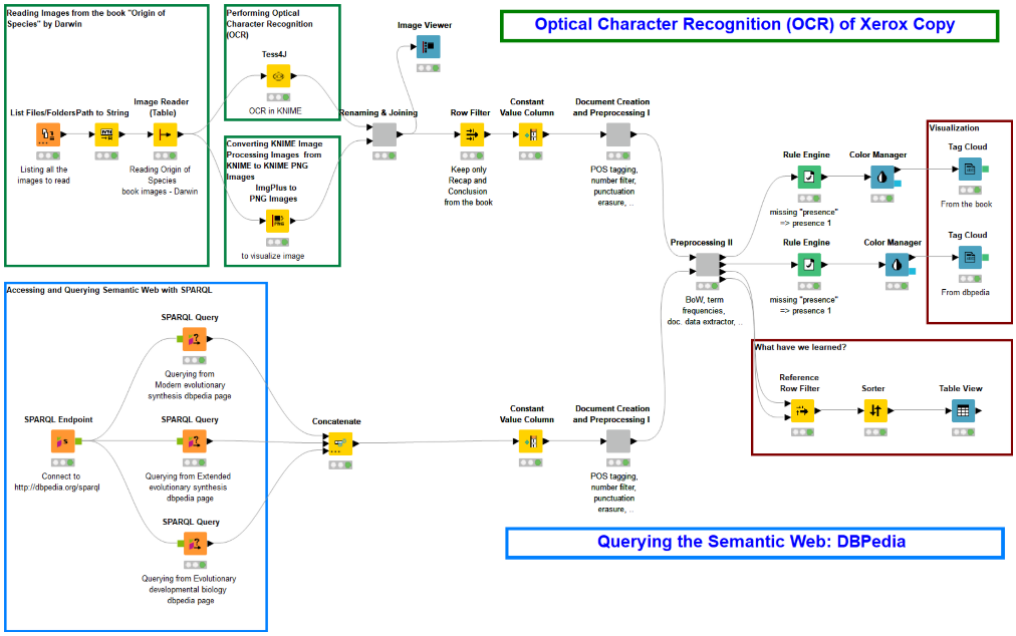
The Image Reader node reads the locally stored image files of the pages of the book "On the Origin of Species".

- 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 the figure below, the part labelled as “Optical Character Recognition of Xerox Copy”.

Querying the Semantic Web: DBPedia



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.

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, 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” component. Here standard text processing functions are applied, such as: case conversion, punctuation removal, number filtering and POS tagging.
- In the following component, 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 Table View.

The Results

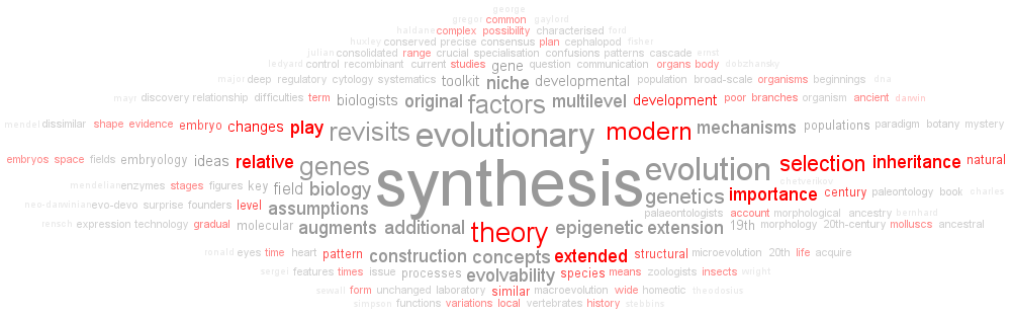
The figures below both show a tag cloud, one with the terms from Darwin’s book “On the Origin of the Species”, the other with the terms found in the results from DBPedia queries.

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 upper tag cloud. 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 lower tag cloud.

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.

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



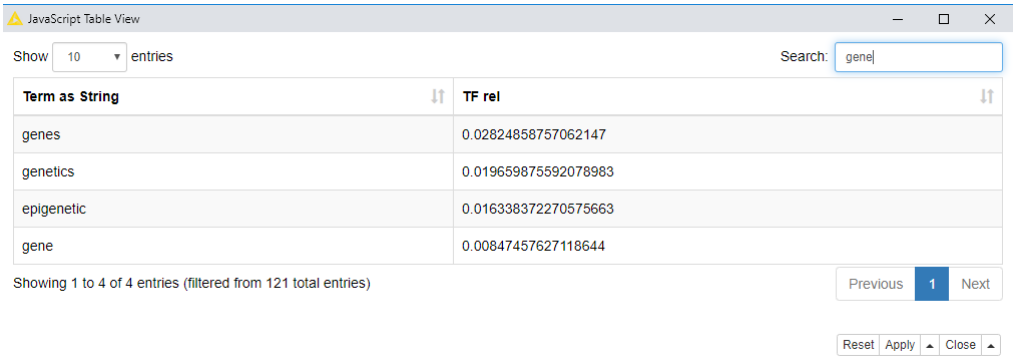
Word Cloud generated from results of DBPedia queries: “Modern evolutionary synthesis”, “Extended evolutionary synthesis”, and “Evolutionary developmental biology”.

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 Table View node (see image below), 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.

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!



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 Table View node in the final workflow.

Public Google Sheets Meets Excel Sheets

A Recipe for Delicious Data

Authors: Lada Rudnitckaia, Anna Martin, Rene Damyon and Oleg Yasnev KNIME

Workflow on KNIME Community Hub: Public Google Sheet Meets Excel Sheet

The Challenge

A local restaurant has been keeping track of its business in 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 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 were built and released in KNIME Analytics Platform 3.5. A number of 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: Access Google Sheets with these new dedicated nodes and mash 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 Google Sheets nodes.

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 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.

Accessing a Google Sheet Document

From a web browser:

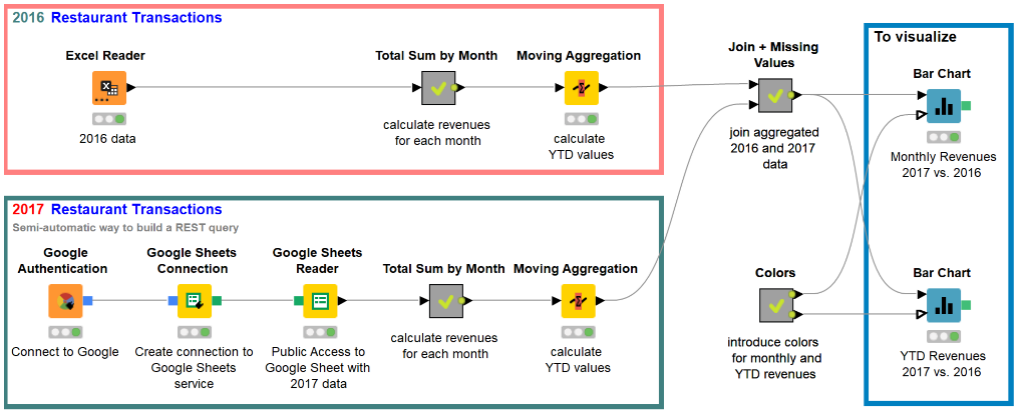
- Create a Google account at www.google.com (if you do not have one already). This account will be used for user authentication to access the Google Sheets document.
- Open the Google Sheets document in a browser while logged into that Google account. This will allow the Google Sheets Reader node to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

Now in the workflow:

- First, we need to connect to Google account store our authorization credentials using the Google Authentication node.
- Next, we need to create connection to Google Sheets service using the Google Sheets Connection node.
- Finally, 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.

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, Public Google Sheet Meets Excel Sheet, to access data from public and private Google Sheets, is shown below and can be downloaded from the KNIME Community Hub.



This workflow blends together data from an Excel spreadsheet and data from a public or private Google Sheet document.

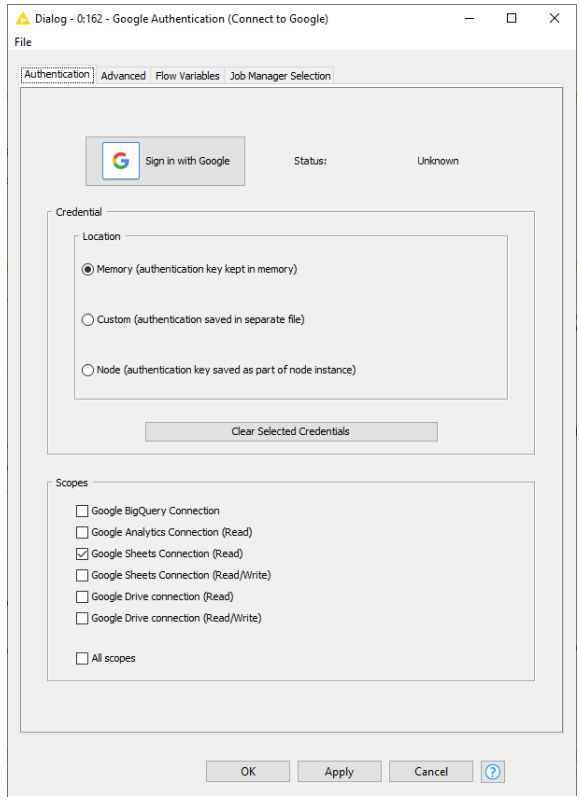
Authentication of Google User

The node to authenticate the Google user is the Google Authentication node. This node connects to Google and authenticates the user via the Google authentication page.

If you open the node configuration window, just click the button named *Sign in with Google*. 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 Status will change from “Unknown” to “Authenticated”.

These authentication credentials can be saved in memory (option *“Memory”*), can be persisted to a separate file to share with other



The configuration dialog for the Google Authentication node.

workflows (option "*Custom*") or they can be saved as a part of node instance (option "*Node*").

> ***Note.*** When exporting your workflow or when in need to make the workflow forget the authentication credentials, just click the button named Clear Selected Credentials.

Connecting to Google Sheets service

Connecting to the Google Sheets service is available with Google Sheets Connection node that doesn't require configuration.

Selecting and Reading the Google Sheet Document

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

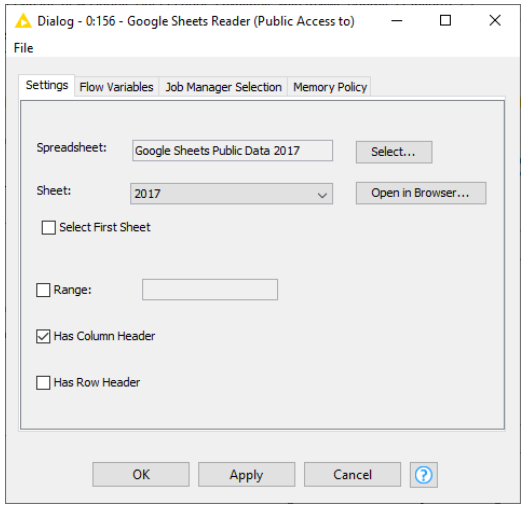
Configuration is as simple as clicking the Select button to display a list of available documents and choosing the one we want. The "Open in Browser" option in the configuration window of the Google Sheets Reader node opens the document on a web browser for a quick inspection.

> ***Note.*** 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 Public Data 2017. These data were simulated using a Gaussian distribution with a mean slightly higher than the mean for data of 2016.

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.

The rest of the workflow calculates the revenues by month as total sum and as



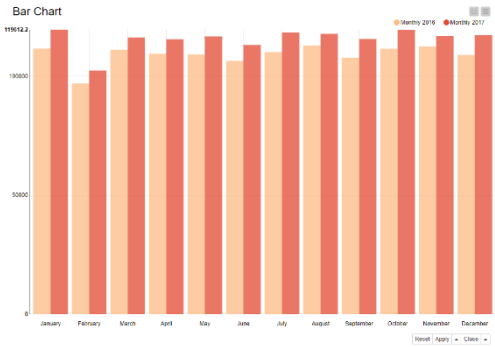
The configuration dialog for the Google Sheets Reader node.

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.

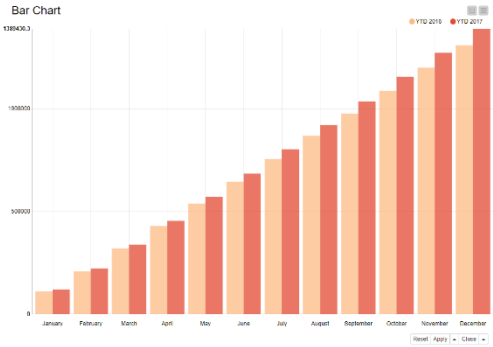
The Results

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 can see that business sales have increased in 2017 with respect to the same months in 2016.

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. 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!



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.



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.

SQL Meets NoSQL

A Database Jam Session

Author: Rosaria Silipo, KNIME

Workflow on KNIME Community Hub: Database Jam Session

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 6. 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”, 6 means “very positive”, and 2, 3, 4 and 5 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:

1. 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 component that uses Credentials Configuration node. The Credentials Configuration 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.
2. 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, Oracle and MS 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 DB Connector node. In this case, the JDBC driver file has to be uploaded to KNIME Analytics Platform via the Preferences -> KNIME -> Databases page. Once the JDBC driver file has been uploaded, it will also appear in the drop-down menu in the configuration window of the DB Connector node. Provided the appropriate JDBC driver is selected and the database hostname and credentials have been set, the DB Connector node is ready to connect to the selected database. Since a dedicated connector node was missing, we used the DB

Connector node to connect to MariaDB databases. Please note that even though there is a dedicated node for Oracle, due to license restrictions the Oracle JDBC driver is not part of the KNIME Analytics Platform and needs to be downloaded and registered separately.

Note. For more details on where to download and how to register a driver see the database documentation.

3. Select Table and Import Data.

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

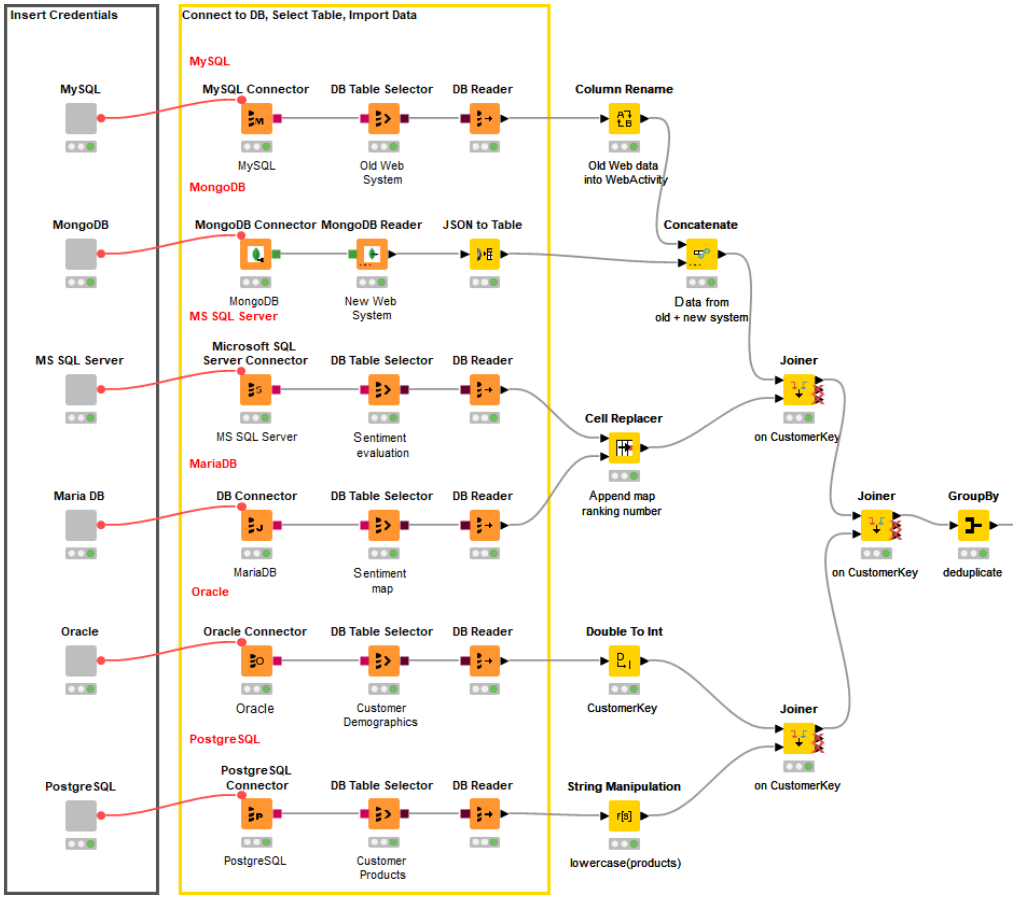
It is comforting that this approach - *connect to database, select table, and extract data* – works for all relational databases. It is equally comforting that the DB Connector node can reach out to any database. This means indeed that with this schema and with the right JDBC driver file I can connect 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.

KNIME MongoDB Integration the nodes that allow you to perform basic database operations on a MongoDB database. In particular, the MongoDB Connector node connects to a MongoDB database; MongoDB Reader node extracts data according to the query defined in its configuration window.

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.



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

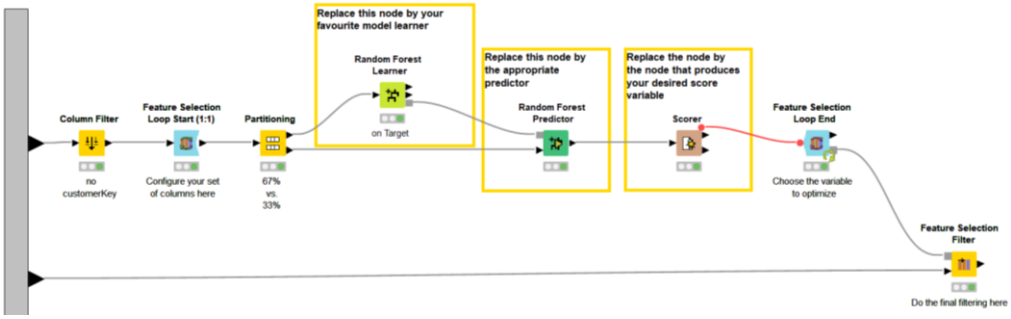
The metanode named "Backward Feature Elimination" and available in the Node Repository under *Analytics/Mining/Feature Selection/Meta nodes*, implements exactly this procedure. The final node in the loop, named "Feature Selection Filter", produces a summary of the model performance, for all steps where 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:

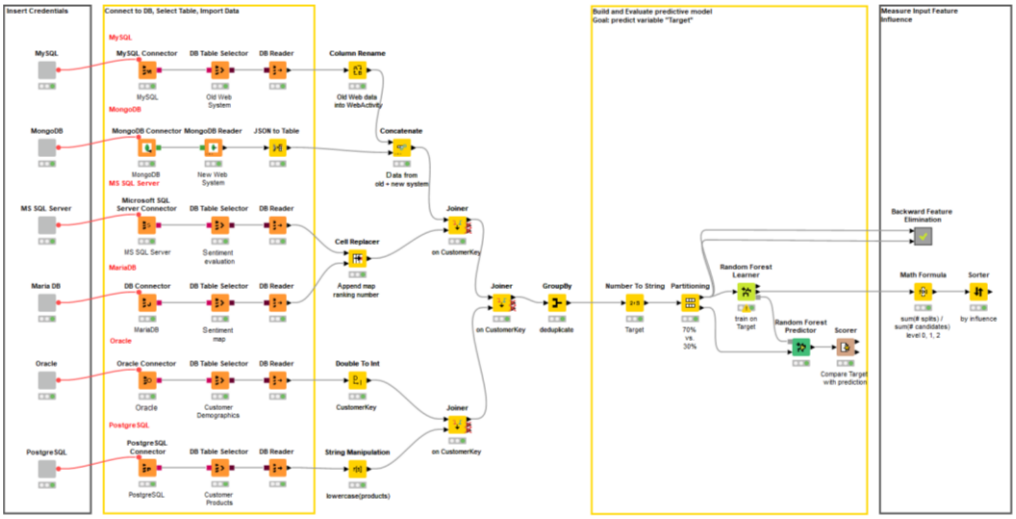
```
influence index = Sum(# splits) / sum(# candidates)
```

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



Content of the metanode "Backward Feature Elimination" adapted for a random forest predictive model.

The final workflow, Database Jam Session, shown below, can be downloaded from the KNIME Community Hub. In the figure 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.



This workflow blends data from 6 different databases: MySQL, MongoDB, SQL Server, MariaDB, Oracle 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.

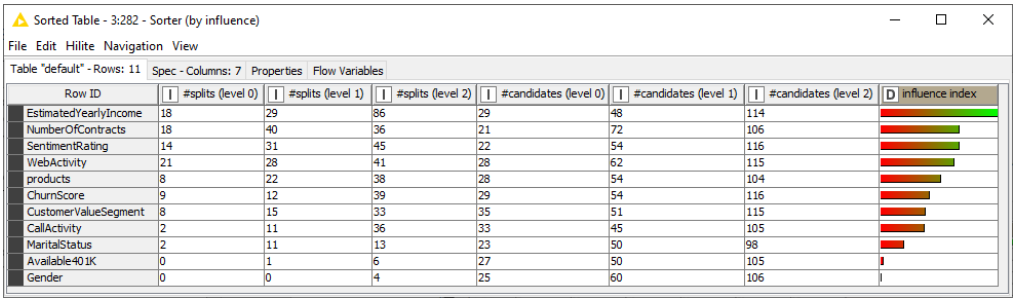
The Results

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, MS SQL Server, and PostgreSQL* – to reach one single data table summarizing all available information about our customers.

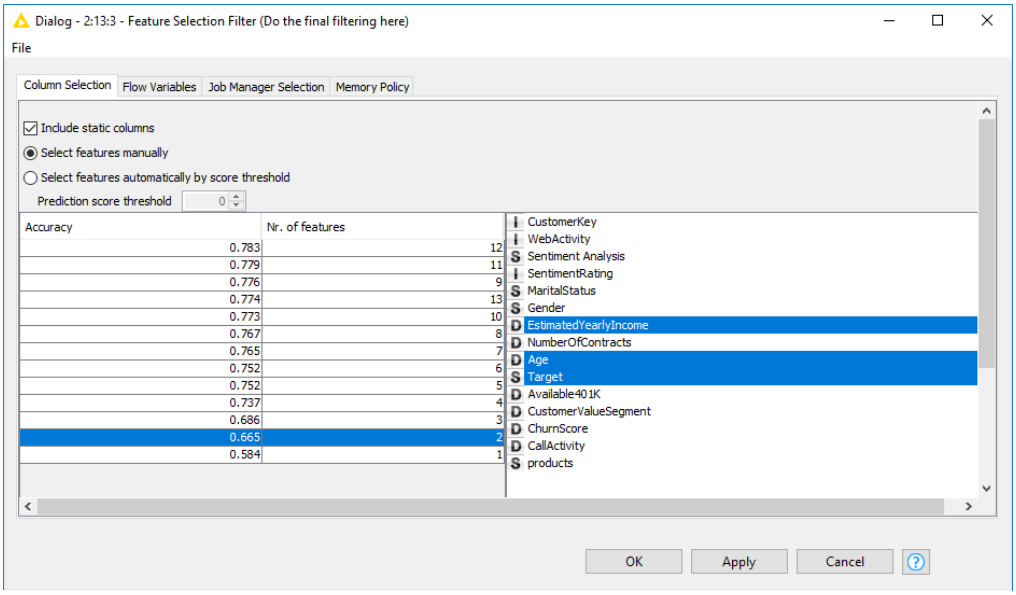
In this same experiment, we also trained a random forest model to predict the likelihood of a customer buying a second product.

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 the figures below. 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.



Row ID	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)	Influence index
EstimatedYearlyIncome	18	29	86	29	48	114	
NumberOfContracts	18	40	36	21	72	106	
SentimentRating	14	31	45	22	54	116	
WebActivity	21	28	41	28	62	115	
products	8	22	38	28	54	104	
ChurnScore	9	12	39	29	54	116	
CustomerValueSegment	8	15	33	35	51	115	
CallActivity	2	11	36	33	45	105	
MaritalStatus	2	11	13	23	50	98	
Available401K	0	1	6	27	50	105	
Gender	0	0	4	25	60	106	

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



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.

SparkSQL Meets HiveQL

Women, Men, and Age in the State of Maine

Authors: Rosaria Silipo and Anna Martin, KNIME

Workflow on KNIME Community Hub: Spark SQL Meets Hive SQL

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 and want to process it on Hive or on Spark using the KNIME Big Data Extension.

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?

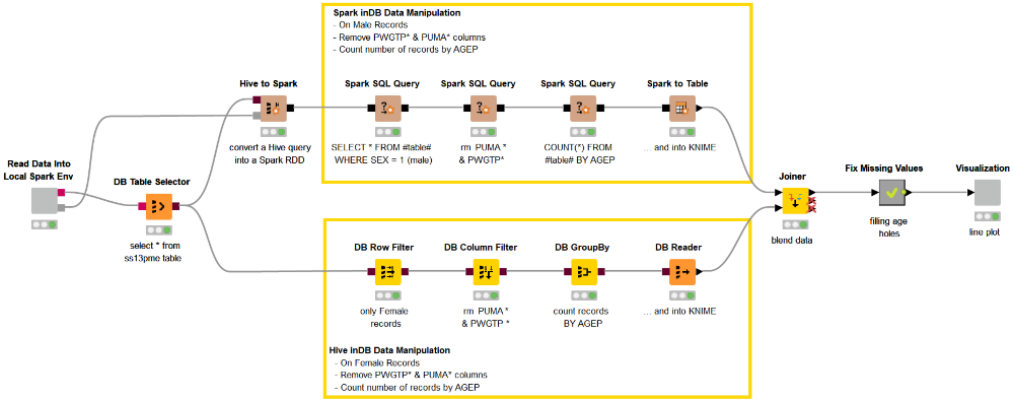
Topic. Age distribution for men and women in the US state of Maine

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.



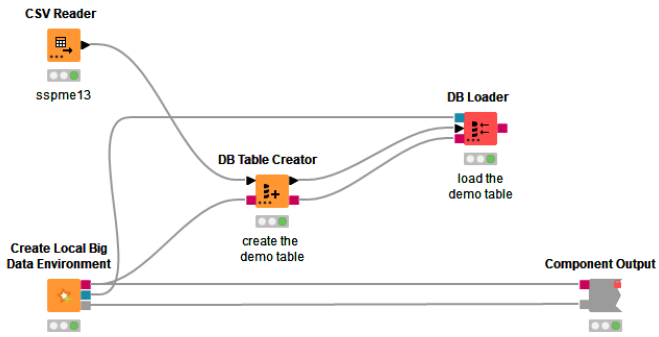
This workflow, Spark SQL Meets Hive SQL, accesses a Hive database, extracts the CENSUS data for Maine residents, processes female records on Hive and male records on Spark, blends the results, and visualizes the two age distributions of men and women. It can be downloaded from the KNIME Community Hub.

Note. If we indeed had our data stored on, e.g., a Hive installation on a Cloudera cdh5.8 cluster running on the Amazon cloud, we would use the dedicated Hive Connector node. However, in this example we create local big data environment with Create Local Big Data Environment node. It is very easy to use if the data is not too big for local space.

Creating local big data environment and demo table

The workflow starts with the component “Read Data Into Local Spark Env” where we first create the local big data environment. We leave the default configuration of the Create Local Big Data Environment node. Note that SQL Support is crossed for HiveQL and provide JDBC connection to support HiveQL syntax and work with a local Hive instance.

File *ss13pme.csv*, provided with the workflow, contains the CENSUS data of 1% of the Maine population over the 5 years following 2009, a total of circa 60k records. We read it with the CSV Reader node and then create the demo table with the DB Table Creator node and load it with the DB Loader node in the local big data environment.



The "Read Data Into Local Spark Env" component.

The component is followed by a DB 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.

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 a Spark context. It is already created by Create Local Big Data Environment node. Then we need to create a Spark DataFrame for the data from Hive. That is achieved by using Hive to Spark node.

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.

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 DB Row Filter node, a DB Column Filter node, and a DB GroupBy node.

At the end, the DB Reader node executes the SQL query on our local big data environment and exports the results into a KNIME data table.

Note. The sequence of DB 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 – DB Reader– that executes the query and gets the job done.

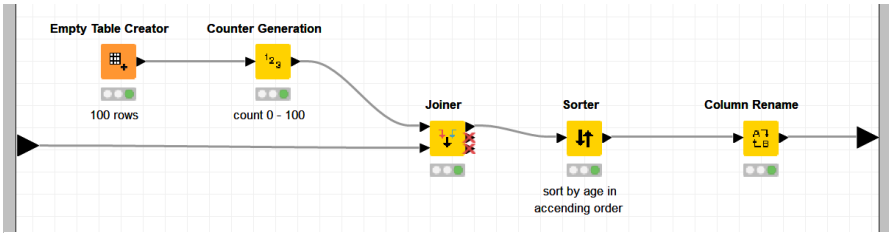
Similarly, the sequence of Spark SQL nodes in the upper branch only builds the sequence of required DataFrames without executing them. It is the final node of the sequence – Spark to Table – that executes the DataFrames and gets the job done.

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, transferring 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 Fix Missing Values metanode.

The final node of the workflow, named Visualization, produces a line plot for both age distributions through a Line Plot node.

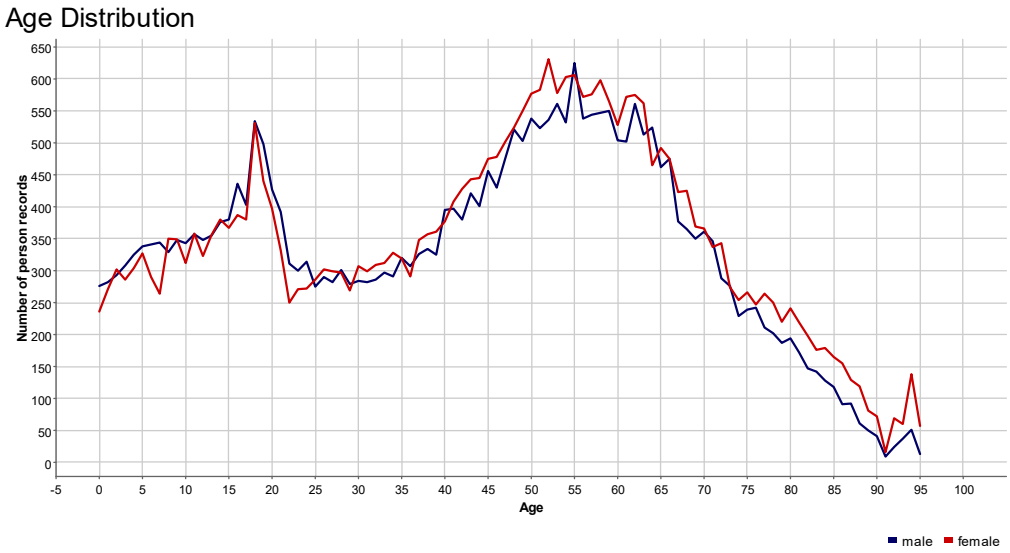
Note. Packing the JavaScript visualization node into a component automatically makes the plots also visible and controllable as a data app from the KNIME Business Hub.

The whole workflow, Spark SQL Meets Hive SQL, is displayed above and is calculating the age distribution for men through Spark SQL and the age distribution for women through Hive SQL.

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 data app interactive page displaying the two age plots is show below. On the y-axis we have the absolute number of people; on the x-axis the corresponding age values.



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!

Chinese Meets English Meets Thai Meets German Meets Italian Meets Arabic Meets Farsi Meets Russian

Around the World in 8 Languages

Authors: Lada Rudnitckaia, Anna Martin, Hayley Smith, and Mallika Bose, KNIME

Workflow on KNIME Community Hub: Blend Languages in a Tag Cloud

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

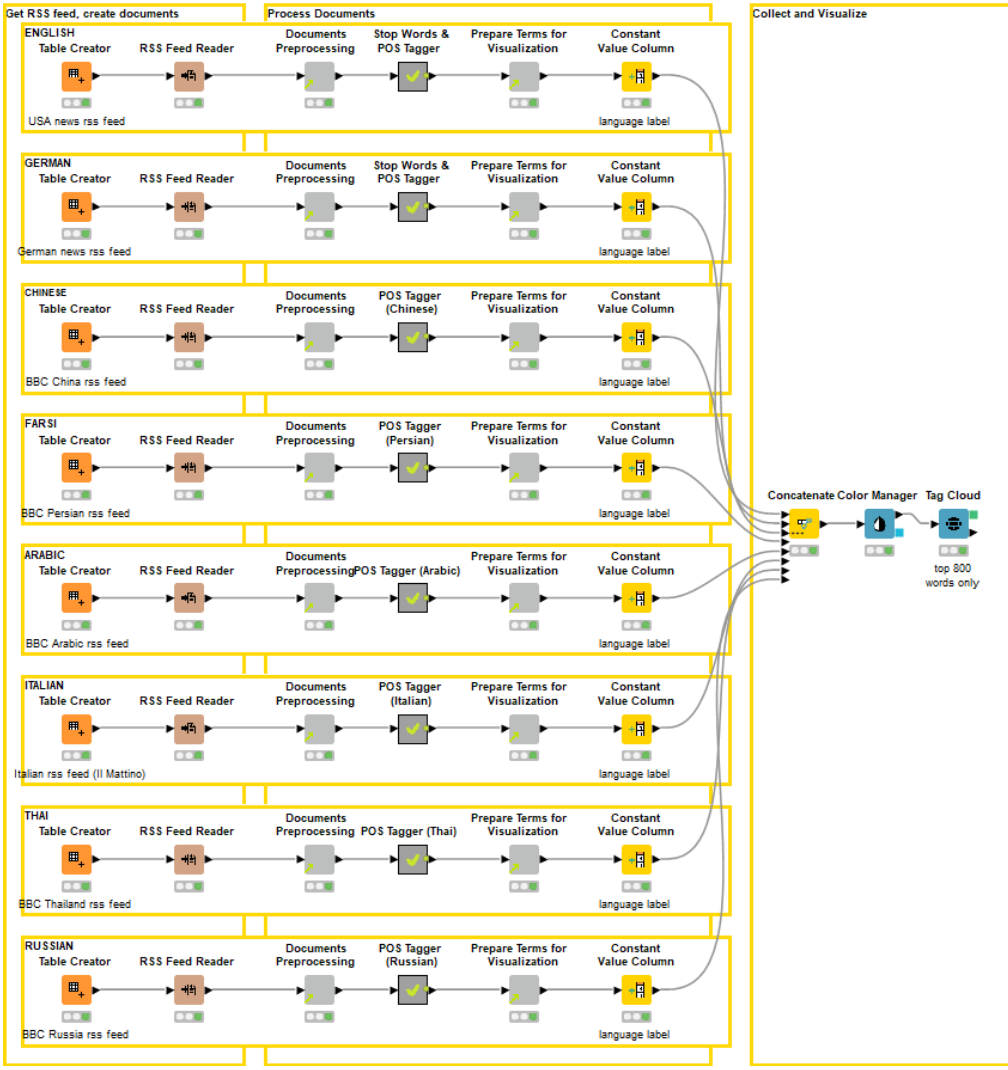
The Experiment

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.

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, Arabic, Turkish 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.



The final workflow is 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. This workflow, Blend Languages in a Tag Cloud, is available on the KNIME Community Hub.

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 component “Documents Preprocessing” can be used for all languages. Not only eight replicas of the same component, but eight links to one single component template!

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

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

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

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

This link to a component template feature was made available in KNIME Analytics Platform version 3.4. The same is applicable for a simple metanode.

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 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, German, 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 Chinese, Farsi, Arabic 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.

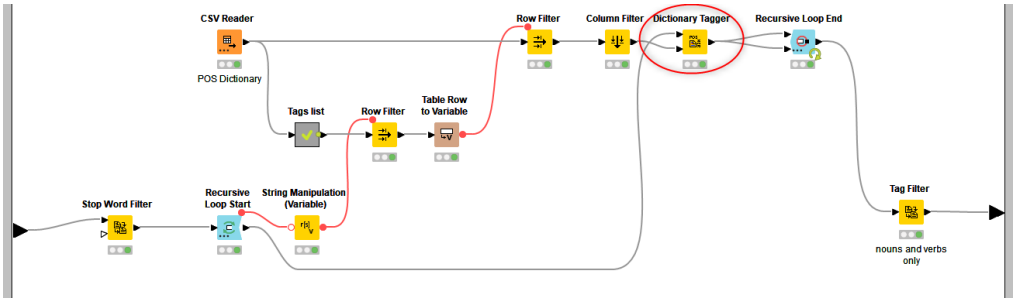
Note. The recursive loop enables looping on the data set for a fixed number of times, or when an end condition is met. Most importantly it has memory, meaning, the output dataset of the current iteration is passed back to be the starting dataset of the next iteration.

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.



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 component, 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. For some languages less than 100 terms are available.

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

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

This workflow, Blend Languages in a Tag Cloud, blends news in eight different languages: English, Italian, German, Chinese, Farsi, Arabic, Thai, and Russian. It is available on the KNIME Community Hub. Notice that the workflow comes in a folder containing also the two component templates.

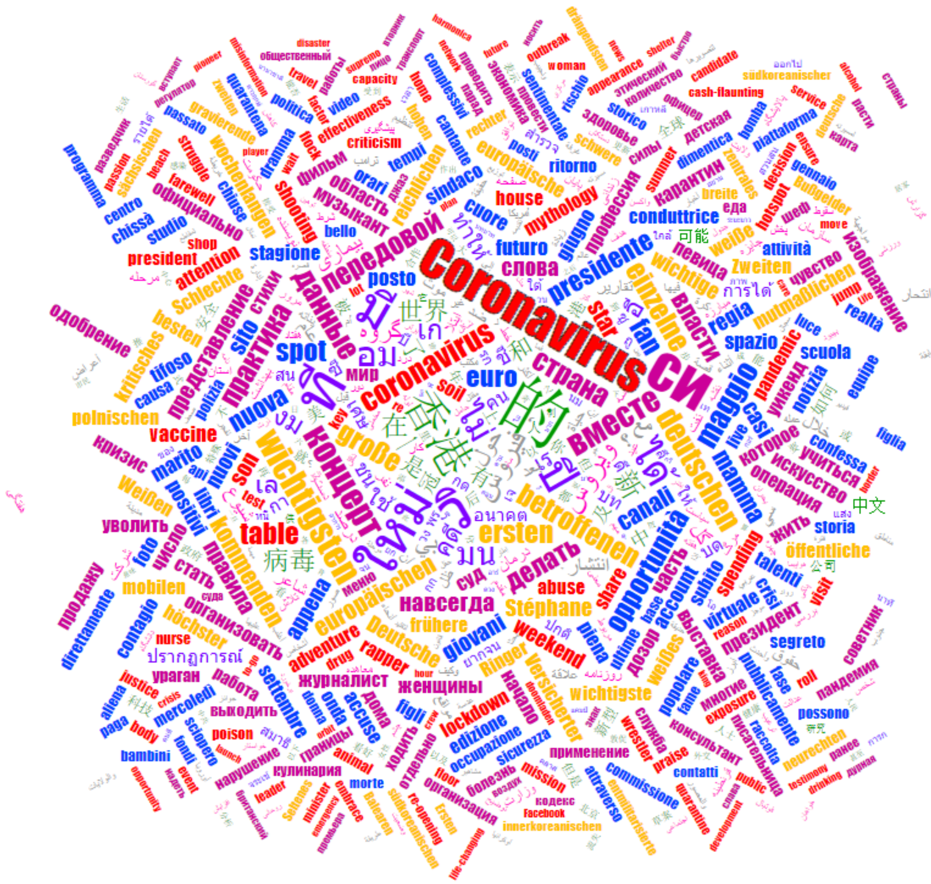
The Results

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

The word cloud view resulting from the Tag Cloud 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.

Wait a second, is it really right time for the world trip? You can see that some frequent words from different languages describe Coronavirus Pandemic: “coronavirus”, “lockdown”, “contagio”, “quarantena”, “病毒”, “пандемия”, “карантин”, etc. This is probably one of the rare situations when different languages do blend in the word cloud. But even if they didn’t, 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.

In general, languages do not blend, but news feeds… yes they blend!



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

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

Custom POS Word Files:

- Chinese http://compling.hss.ntu.edu.sg/omw/
- Farsi 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 that 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.

Finnish Meets Italian & Portuguese through Google Translate API

Preventing Weather from Getting Lost in Translation.

Author: Heather Fyson and Phil Winters, KNIME

Workflow on KNIME Community Hub: Translate using Google API

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

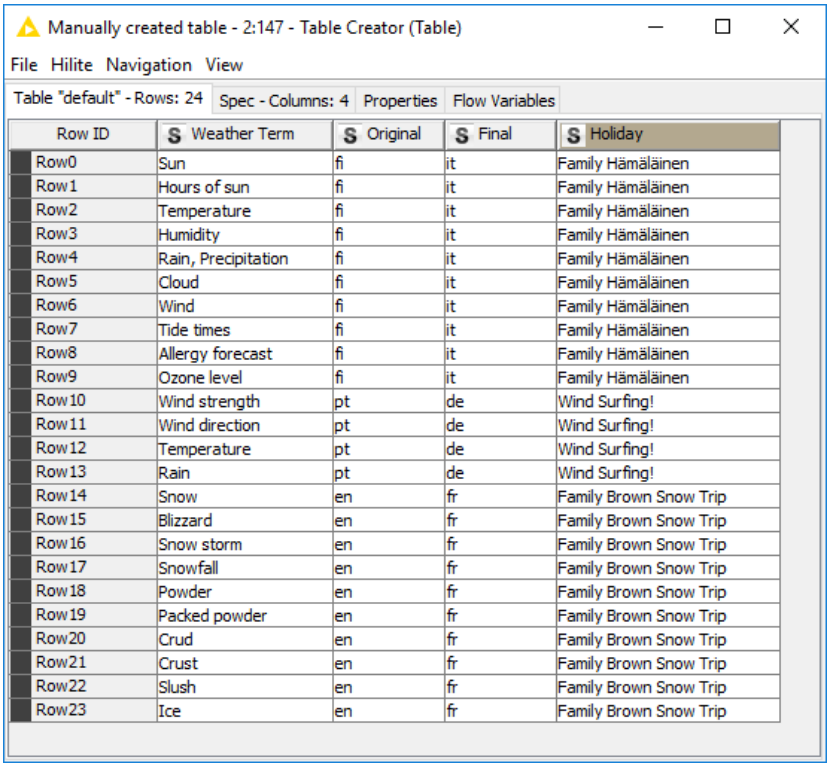
1. Get a Google API key enabled for the Google Translate service from the Google API Console. Here is some help on how to obtain a Google API key. To understand how to use the Google Translate URL, which options are available, and how to append the API key, check the Google Translate API instructions.

   Note. 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.



Manually created table - 2:147 - Table Creator (Table)

File Hilite Navigation View

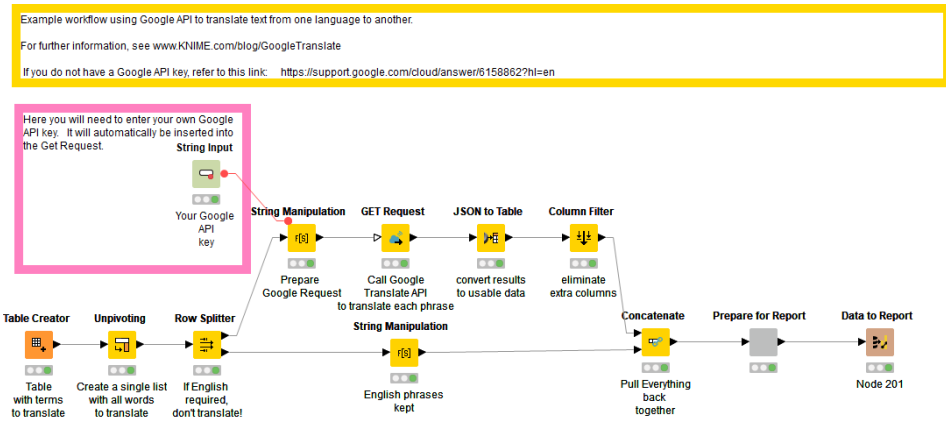
Table "default" - Rows: 24 | Spec - Columns: 4 | Properties | Flow Variables

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

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

1. 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.
2. In the following String Manipulation node, Google Translate Requests are built that combine their Google API Key along with the required options.
3. 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.
4. 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.
5. The English terms, that skipped translation, are added to the end of the table so that we have a complete set of translated terms.

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



This workflow, [Translate using Google API](), accesses the Google Translate API service and is available on the KNIME Community Hub.

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.

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!

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

Weather Card by Winter & Sun Travel agency for the Brown family, created with the Translate using Google API workflow.

YouTube Metadata Meets WebLog Files

What Will it be Tonight – a Movie or a Book?

Author: Rosaria Silipo, KNIME

Workflows on KNIME Community Hub: Apache Logfile Analytics, How to access YouTube REST API

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.

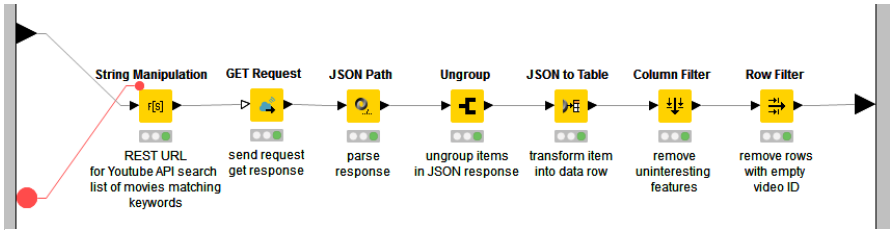
- The service retrieving the comments to the video, named "***commentThreads***":

  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.



The sub-workflow in the "YouTube API..." 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 node. The 2 String Input nodes are contained in the first component, named "Search Keywords & 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 45 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.

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

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

5. The word cloud image is then exported into the report connected to this workflow.
6. 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.

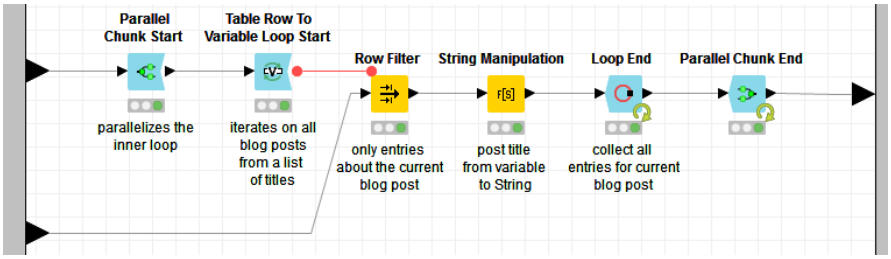
1. KNIME Analytics Platform offers a dedicated node to read and parse weblog files: the Web Log Reader node. The lower workflow branch starts with a Web Log Reader node. In the configuration window of the Web Log Reader node, a few useful parsing parameters can be defined. Besides the file path, which 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 component 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 Table Row 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, 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.



The content of “Extract Post Entries” metanode.

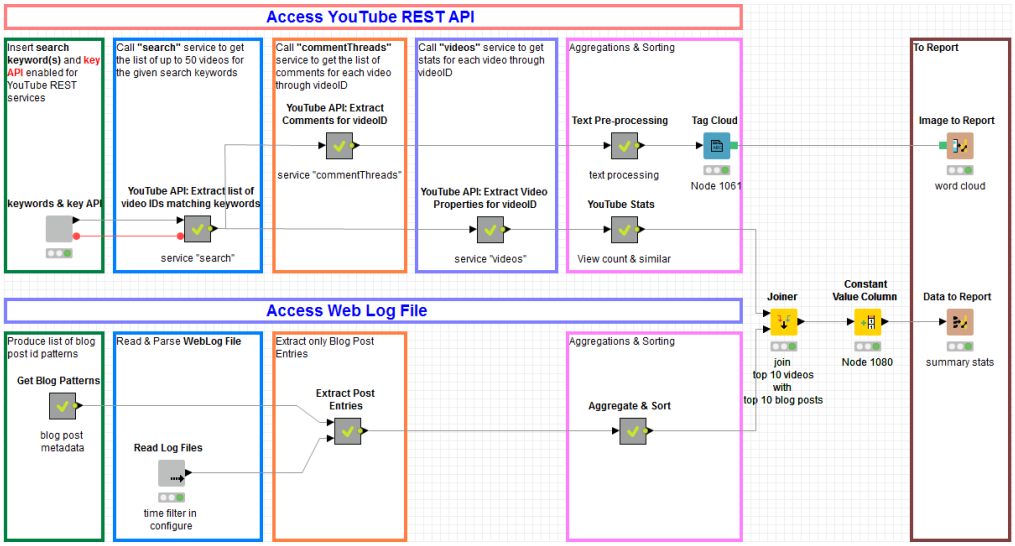
Data Blending and Final Report

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 below.



The final workflow. The upper branch connects to YouTube REST API, the lower branch parses weblog file.

For privacy reasons, we could not make this workflow available as it is in the KNIME EXAMPLES space. However, you can find an example workflow for the WebLog Reader node here. The upper part of this workflow can be found here. 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. 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 (as of March 2017):

- "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

The top 10 KNIME blog posts are:

- "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 sentiment analysis, 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 started in 2012, while the KNIME blog only 2 years later – in 2014. Since we have not set any time limits, the number of views is 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?

Note. 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.

The final Report as PDF document. At page 2 and 3, we find 2 bar charts with the number of views for the topmost 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.

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.

Kindle EPUB Meets Image JPEG

Will KNIME make Peace between the Capulets and the Montagues?

Author: Heather Fyson and Kilian Thiel, KNIME

Workflow on KNIME Community Hub: Will they blend? EPUB Meets JPG - Romeo meets Juliet

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 Julidet, 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.

We downloaded the "Romeo and Juliet" e-book for free 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 component "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 component "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 component "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 component “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

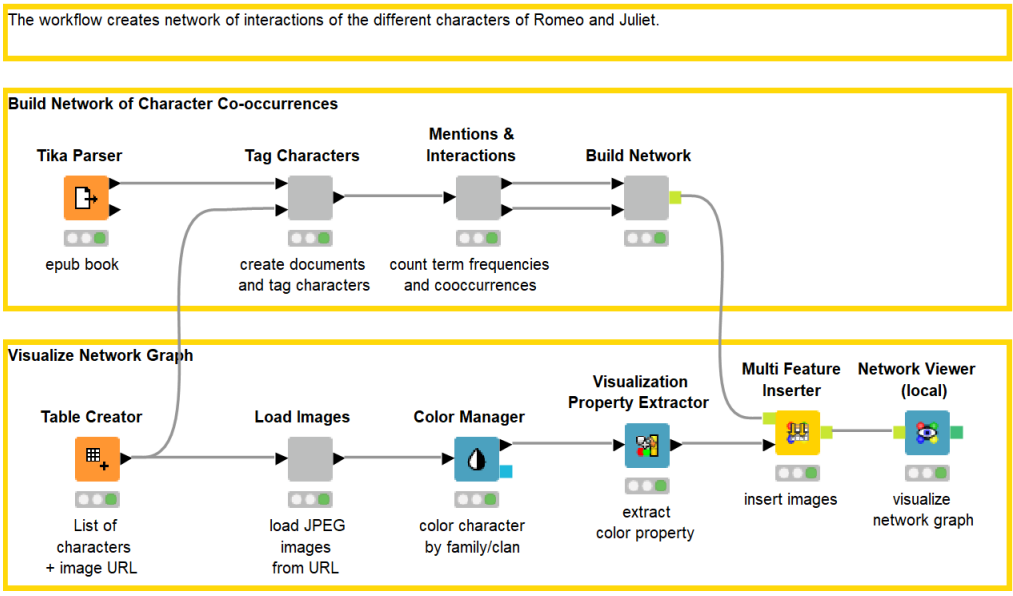
1. JPEG images of the play’s characters are stored locally on the hard disk of the machine. The URL to the image file for each character is manually inserted in a Table Creator node together with the reference character. The Table Creator node allows the user to manually input data in an Excel-like fashion.
2. The list of image URLs feeds the component “Load Images”. Inside this component, 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 component, takes care of this task. The output of the component 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, Will they blend? EPUB Meets JPG - Romeo meets Juliet, is shown below and is available on the KNIME Community Hub.

The final graph showing the interaction degree between the different characters of the play.



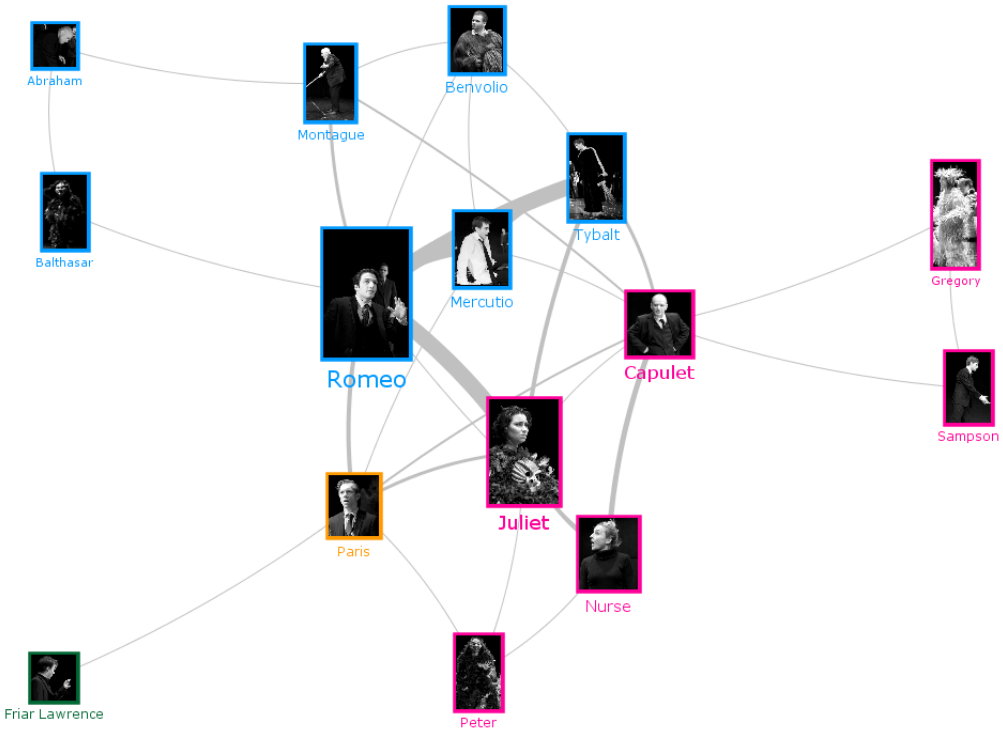
This workflow successfully blends a Kindle epub file of the play "Romeo and Juliet" with the JPEG images of the play's characters in a live show in a dialog graph.

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 blue are on one side; all Capulets in pink 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!



The 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.

SAS, SPSS, and MATLAB Meet Amazon S3

Setting the Past Free

Author: Phil Winters, CIAgenda

Workflow on KNIME Community Hub: SAS, SPSS, and MATLAB Meet S3

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 require 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

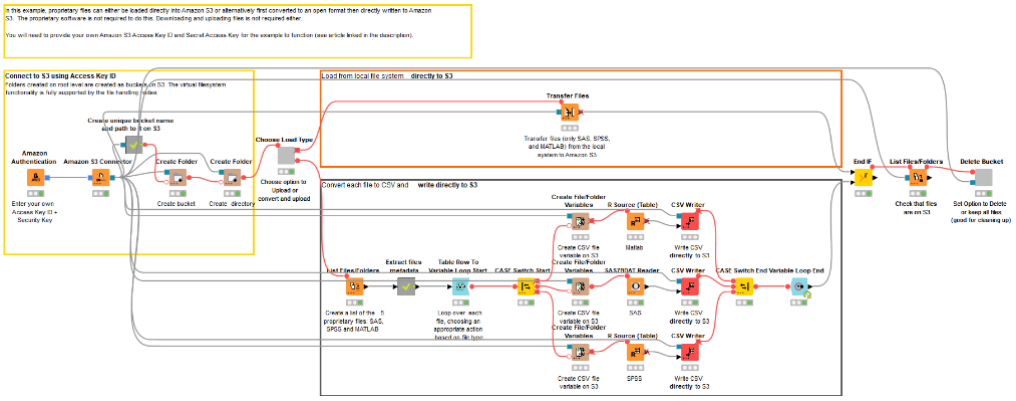
The Setup

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: https://aws.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



Workflow to blend data from SAS, SPSS, and MATLAB platforms and to write them on an Amazon S3 data lake, either in a proprietary or in an open format

In the yellow workflow annotation, I first enter my Amazon S3 credentials into the Amazon Authentication node and provide a working directory in the Amazon S3 Connector 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 yellow workflow annotation 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 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 make that choice with the small component named "Choose Load Type". Simply by right clicking the component, I get a specific configuration window (made with Single Selection Configuration node) that allows me to choose the load type.

If I choose "Proprietary", then the top branch is chosen, where Transfer Files node does a bulk-load of the files filtered by extensions from local file system 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)", then the bottom branch is chosen. Here, List Files/Folders node creates a list of files filtered by extensions. Then, I transform relative paths created by the List Files/Folders node into full local paths to read the files in R and extract some other metadata from the paths. That is what I do in the small metanode named "Extract files metadata".

Then, for each of the proprietary files, I create a path on S3 using Create File/Folder Variables and read each file with the appropriate KNIME node. For each file, once the path is created and the file is read into KNIME table, I write it directly onto S3 using CSV Writer node. Note the LOOP and CASE SWITCH construct, which takes each row and - based on the file type - automatically uses the appropriate READ 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.

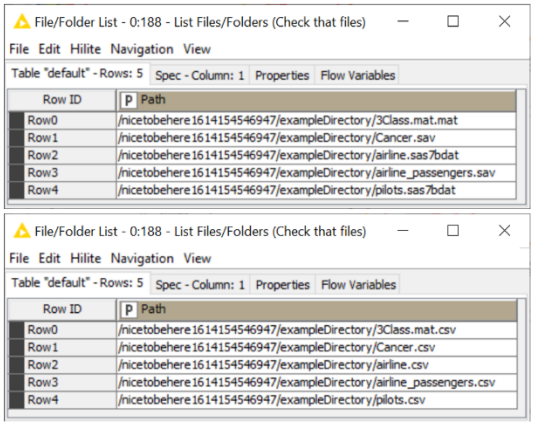
At the end of the flow, I apply the List Files/Folders node to list files located remotely on Amazon S3.

As a very last step, I have a small component 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 need to reset the workflow each time 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 proprietary files (upper) or .csv files (lower), depending on your choice, in the list of remote files from the cloud.



The blended data in proprietary files (upper) or .csv files (lower) on the Amazon data lake.

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 Authentication and Amazon S3 Connector nodes to an Microsoft Authentication and Azure Blob Storage Connector nodes, 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.

In KNIME Analytics Platform you can create streamed components that subset and manipulate your data “on the fly” creating fantastic opportunities to efficiently use KNIMEsoftware on those ever growing data lakes within the cloud.

Note. If you use a supported **streaming** data type, such as CSV or a 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 space on the KNIME Community Hub with the keyword S3 or Azure.

This workflow, SAS, SPSS, and MATLAB Meet S3, is available, without the API keys (of course), on the KNIME Community Hub.

XML Meets JSON

Some News come in XML, some in JSON Format

Author: Rosaria Silipo, KNIME

Workflow on KNIME Community Hub: Will they blend? XML meets JSON

The Challenge

Do you remember the post about blending news headlines from IBM Watson Discovery News and Google News services? (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 JSON structured data – the response from IBM Watson Discovery 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 queried IBM Watson Discovery News service for the first 10 news headlines on Barack Obama and the first 100 news headlines on Michelle Obama. The response for Barack Obama were received in XML format and the response for Michelle Obama in JSON format. Two datasets: 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.

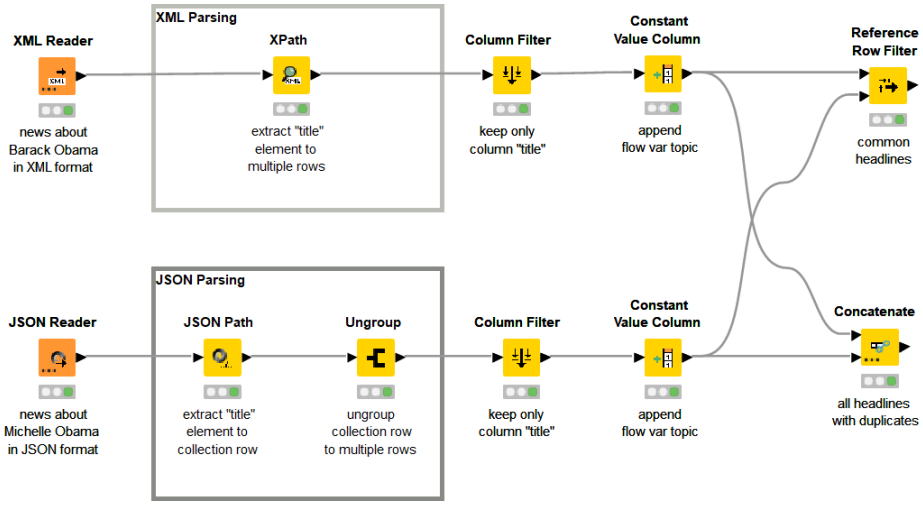
Challenge. Blend the two data response datasets respectively in XML and JSON format.

Access Mode. JSON and XML parsing.

The Experiment

1. The queried data are saved locally and provided with the workflow. We simply read the headlines on Barack Obama with XML Reader node and headlines on Michelle Obama with JSON Reader node.
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 below. A version of this workflow, Will they blend? XML meets JSON, reading IBM Watson responses from a file is on the KNIME Community Hub.



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!

Google BigQuery Meets SQLite

The Business of Baseball Games

Author: Dorottya Kiss, EPAM

Workflow on KNIME Community Hub: Google BigQuery Meets SQLite. The Business of Baseball Games

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 paper 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 BigQuery Public data set and stored them on our own Google BigQuery 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 BigQuery 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 BigQuery and weather data from SQLite.

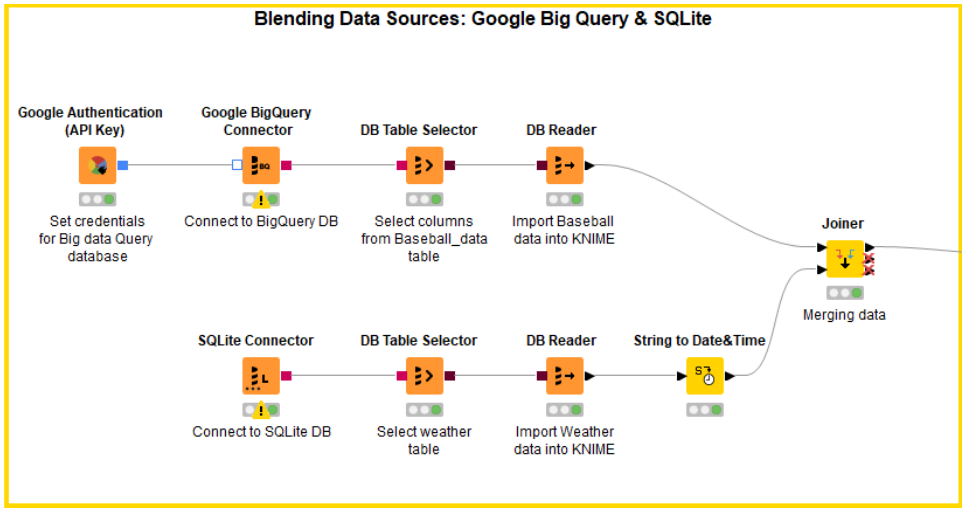
Access Mode: KNIME Google BigQuery Integration and dedicated SQLite Connector node.

The Experiment

The first part of the workflow connects with the two data storage platforms: Google BigQuery 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 both BigQuery and SQLite. Currently, the JDBC driver for Google BigQuery isn't one of the default JDBC drivers, so you will have to add it to KNIME. We will then use the dedicated SQLite Connector node to access SQLite and a dedicated Google BigQuery Connector to connect to Google BigQuery.



First part of the workflow merging Google BigQuery and SQLite data.

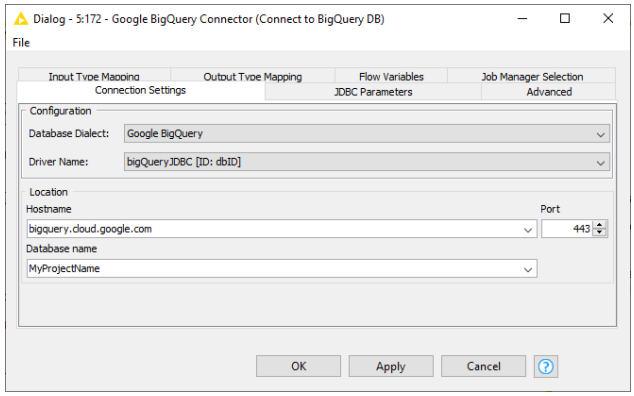
Connect to Google BigQuery

You will first need to to grant access to Google BigQuery and register the JDBC driver in KNIME. You will find a very detailed guidance in this tutorial. In this blog post, we jump to building our KNIME workflow at once.

We start with Google Authentication (API Key) node. To configure the node, we will need the service account ID, in the form of an email address, which was automatically generated when the service account was created and the P12 key file.

We can now connect to the database using the dedicated Google BigQuery Connector node. To configure this node:

1. 1. Under “Driver Name” select the JDBC driver, i.e., the one we named BigQueryJDBC.
2. Provide the hostname, in this case we’ll use *bigquery.cloud.google.com*, and the database name. As database name here, use the project name you created/selected on the Google Cloud Platform.
3. Click OK to confirm these settings and close the window.



The configuration window of the Google BigQuery Connector node.

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

```
SELECT DATETIME(startTime, "UTC") as startTime, gameStatus, attendance, dayNight, homeTeamName, awayTeamName, venueCapacity, venueCity, venueState, awayFinalHits, awayFinalRuns, homeFinalHits, homeFinalRuns

FROM `bigquery-public-data.baseball.games_wide`

GROUP BY startTime, gameStatus, attendance, dayNight, homeTeamName, awayTeamName, venueCapacity, venueCity, venueState, awayFinalRuns, awayFinalHits, homeFinalHits, homeFinalRuns
```

Note. When typing SQL statements directly, make sure you use the specific quotation marks (``) required by BigQuery.

Finally, a DB 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 of the workflow.

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 BigQuery database (see lower branch in the above workflow snippet). Unlike Google BigQuery, SQLite enjoys the privilege of a driver file that is already embedded in the dedicated connector node.

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

After that, a DB Table Selector node points to the Weather table through either manual selection of the table or according to a quite simple custom SQL query:

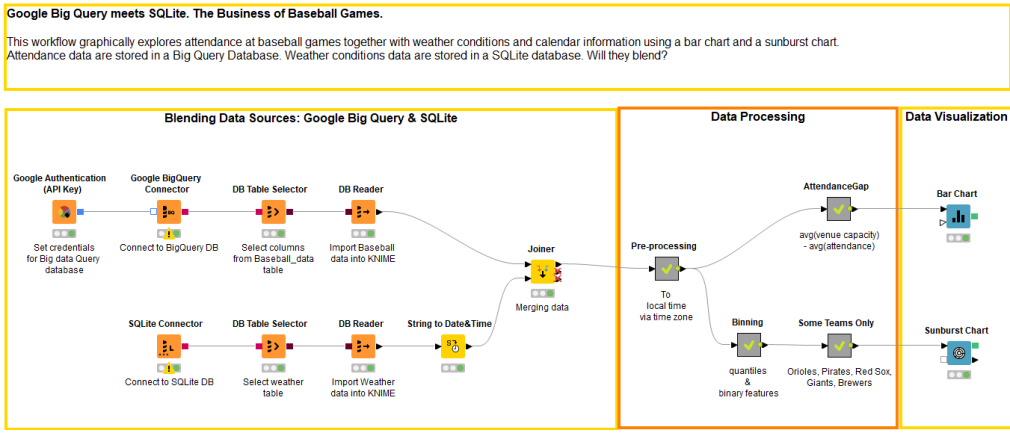
```
SELECT * FROM Weather
```

Finally, a DB 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 DB Table Selector node, the Select a Table button in the top right corner can come to the rescue. Indeed, this button allows you to explore the database content in the Database Metadata Browser.

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 BigQuery. The String To Date&Time node converts the dates imported from SQLite into KNIME Date&Time type. 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 below.



The final workflow merges attendance data from Google BigQuery and weather data from SQLite, pre-processes the resulting dataset, and visualizes attendance and the influence of the weather on attendance. This workflow, Google BigQuery Meets SQLite. The Business of Baseball Games, is available on the KNIME Community Hub.

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.

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 with a 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.

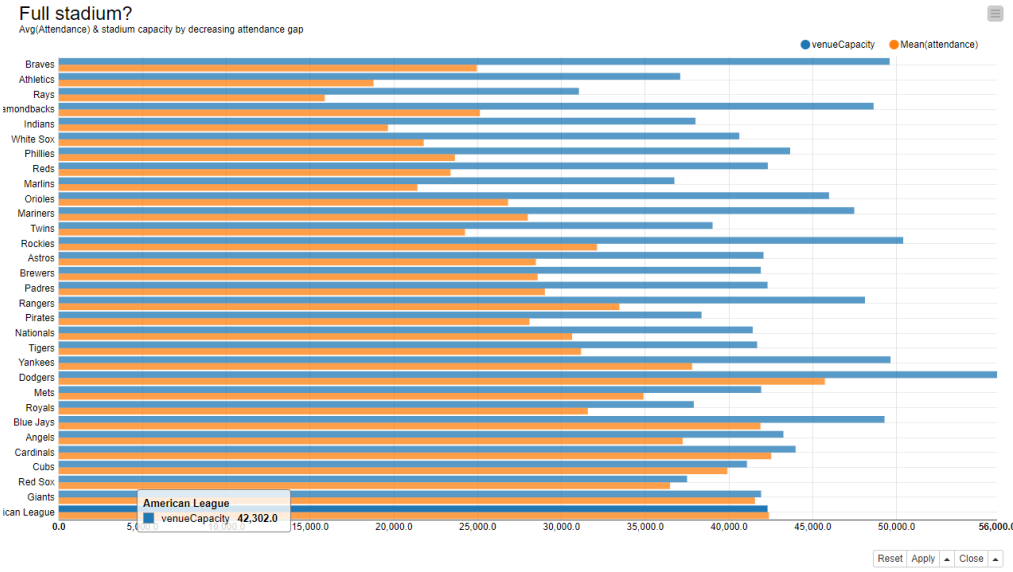
In order to make the final chart easier to inspect, we concentrate on only 5 home teams: Orioles, Pirates, Giants, Red Sox, and Brewers. The Sunburst Chart node ends this branch of the workflow.

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.

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.



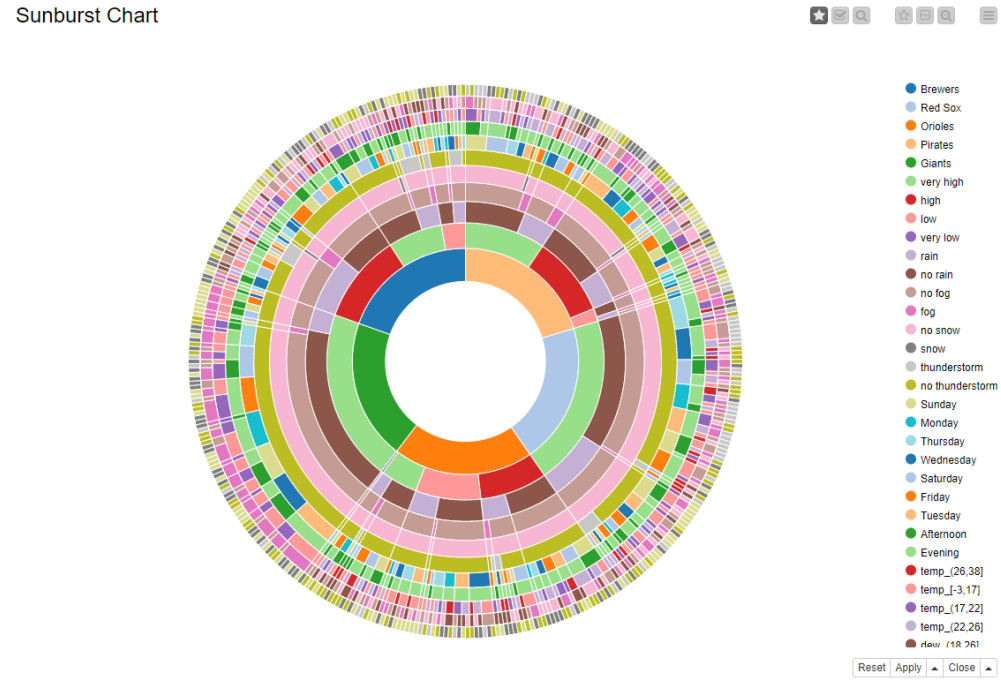
Bar Chart of average attendance and stadium capacity by home team sorted by attendance gap ratio.

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 the figure 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.

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.

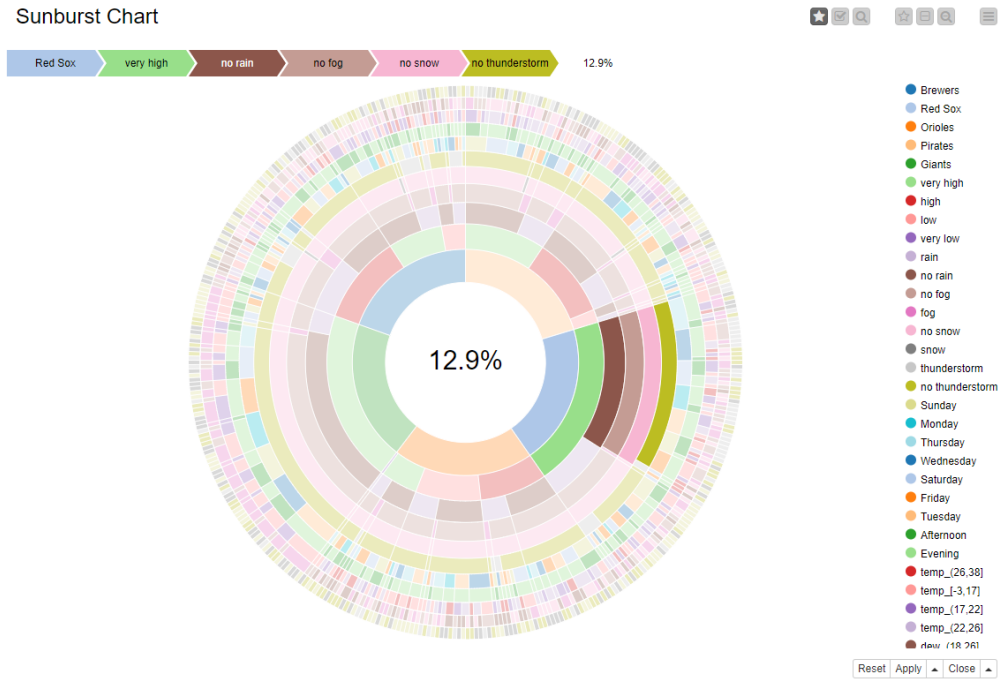
For example, in perfect weather conditions (no rain, no snow, no thunderstorm, no fog), we have a very high attendance for the Red Sox. This segment accounts for almost 13% of the records in the dataset. If you inspect the chart further in focus mode, however, you discover that the Red Sox always have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



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

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.



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.

This workflow, Google BigQuery Meets SQLite. The Business of Baseball Games, is available on the KNIME Community Hub.

MS Access Meets H2

Test your Baseball Knowledge

Author: Vincenzo Tursi, KNIME

Workflow on KNIME Community Hub: Will they blend? MS Access - H2

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 hitting 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 Business Hub, 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 Business Hub.

The Experiment

Accessing MS Access database

1. To access the MS Access database we rely on the Microsoft Access Connector node.
2. Then, two DB Table Selector nodes select the Salaries table and the Master table, containing respectively the players' salaries and demographics information.
3. 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.

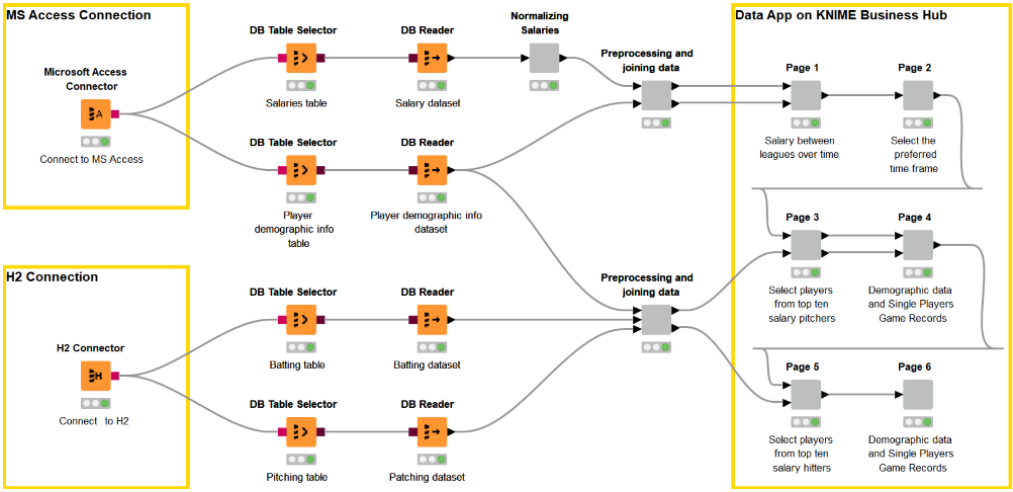
Blending data from MS Access and H2 databases

After some transformation and cleaning in "Normalizing Salaries" component and two "Preprocessing and joining data" components, such as computation and normalization

of the average stats for each player, we created a sequence of 6 webpages to display in a data app on the KNIME Business Hub:

1. **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.
2. **Page 2** allows the user to select a 5-year time frame between 1985 and 2015.
3. **Page 3** displays the top 10 best paid pitchers for the selected time frame in both National and American League and allows you to select some of them.
4. **Page 4** numerically describes the selected pitchers through average statistics and radar plots.
5. **Page 5** displays the top 10 best paid hitters for the selected time frame in both National and American League and allows you to select some of them.
6. **Page 6** numerically describes the selected hitters through average statistics and radar plots.

The final workflow, Will they blend? MS Access - H2, is available for on the KNIME Community Hub.



This workflow blends data from MS Access database and H2 database, on the left. After blending, the last 6 components on the right implement 6 data app pages to guide data selection and description on the KNIME Business Hub.

Note. For any other database, just change the connector node. In the category, Database/Connection in the Node Repository, a number of dedicated connector nodes are available. If you cannot find yours, you can always resolve to the generic DB Connector node. The DB 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 of the data app shows the growth of the average salary over the years. That is quite some growth! Was it worth it?

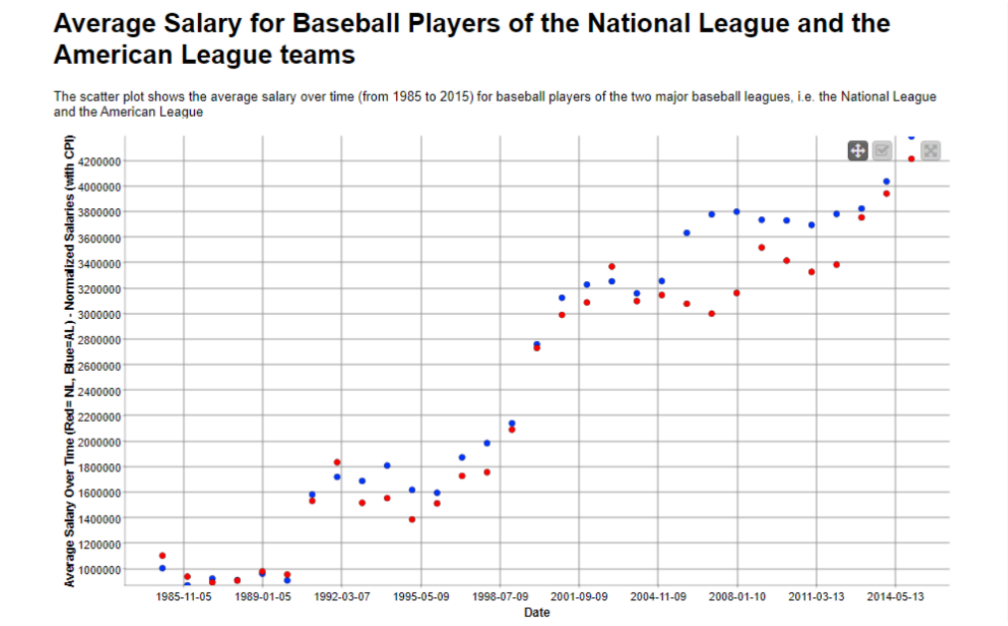
Let's select a time frame on page 2, for example 1985 to 1990.

On page 3, let's select 2 of the top 10 best paid pitchers active between 1985 and 1990: Frank Viola and Orel Hershiser. In the figure extracted from page 4, you can see that Orel Hershiser and Frank Viola indeed exhibited fantastic pitching stats.

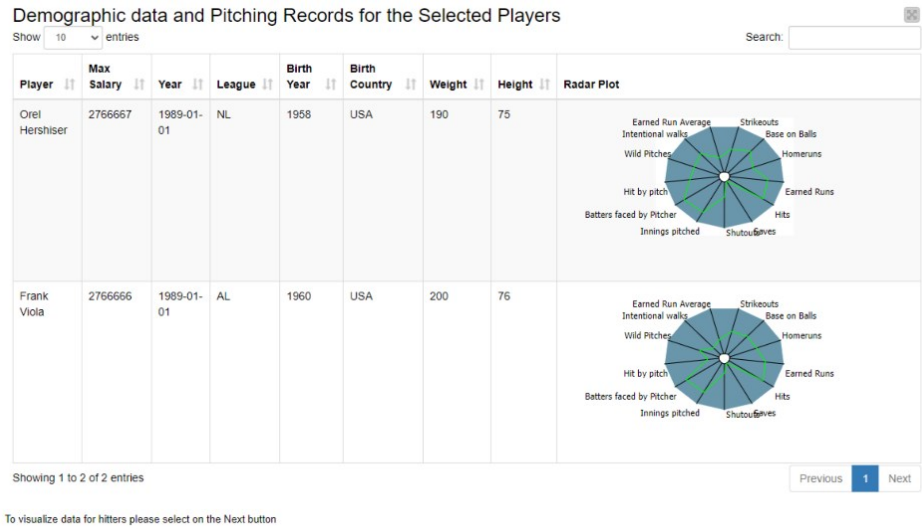
On page 5, let's select 2 players from top 10 best paid hitters active between 1985 and 1990: George Foster and Orel Hershiser. In the figure extracted from page 6, you can see that George Foster and Orel Hershiser indeed exhibited fantastic hitting stats.

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

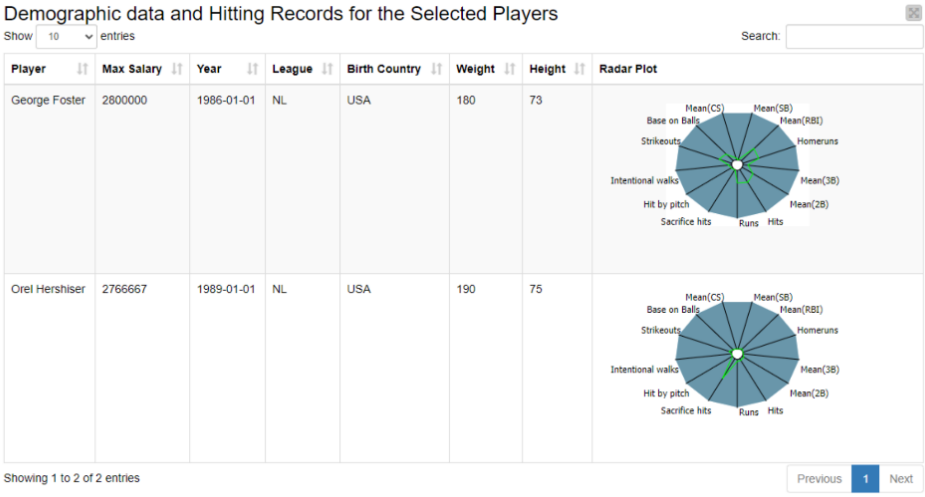
This data app is the proof of the successful blending of data from MS Access and H2 database. The most important conclusion of this experiment is again: Yes, they blend!



Page 1 of the data app on the Business Hub. This scatter plot shows the growth of the average salary in baseball from 1985 to 2015.



Page 4 of the data app on the Business Hub. The table shows demographic data and pitching stats for the selected players.



Page 6 of the data app on the Business Hub. The table shows demographic data and hitting stats for the selected players.

Amazon S3 Meets MS Azure Blob Storage

A Match Made in the Clouds

Author: Rosaria Silipo, KNIME

Workflow on KNIME Community Hub: Will They Blend? Amazon S3 Meets MS Blob Storage plus Excel

The Challenge

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

Let's suppose now that because of 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 Authentication and Connector nodes; respectively the Amazon Authentication and the Amazon S3 Connector node for one cloud service and the Microsoft Authentication and the Azure Blob Storage Connector nodes for the other cloud service.

1. From Amazon S3
    a. First, the Amazon Authentication node connects to the Amazon S3 service using the credentials provided in its configuration window.
    b. Next, in the configuration window of the Amazon S3 Connector node we can provide the working directory of any depth and browse folders on Amazon S3 just like on a local file system.
    c. Finally, in the configuration window of the CSV Reader node with the activated dynamic port, we can browse the S3 folders, select ss13hme.csv, and read it into the KNIME table directly from the Amazon S3 platform.
2. From MS Azure Blob Storage (same procedure with dedicated nodes as for Amazon S3)
    a. First, the Microsoft Authentication node connects to the Azure Blob Storage service using the credentials provided in its configuration window.
    b. Next, in the configuration window of the Azure Blob Storage Connector node we can provide the working directory of any depth and browse folders on Azure Blob Storage.
    c. Finally, in the configuration window of the CSV Reader node with the activated dynamic port, we can browse the Azure Blob Storage folders, select ss13pme.csv, and read it into the KNIME table directly from the Azure Blob Storage platform.
3. Now both datasets 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 datasets 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;

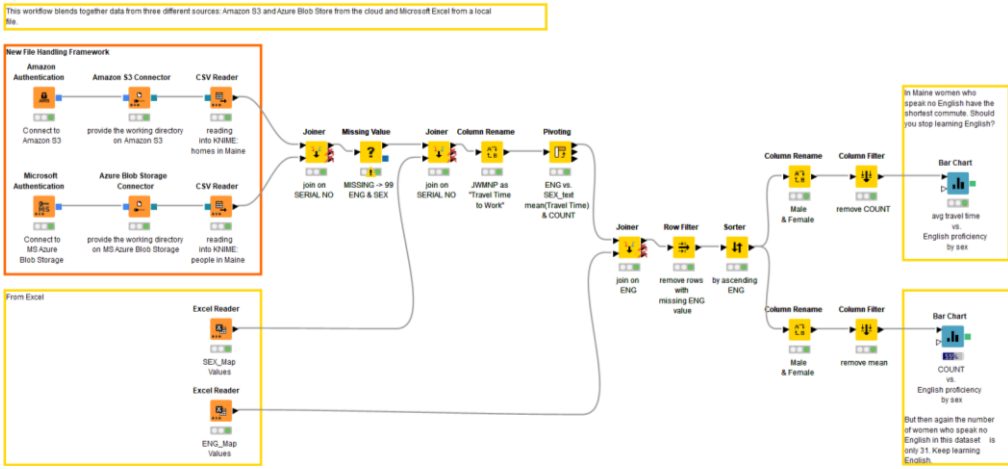
and JWMNP contains the travel time to work. The idea is to see if there is any interesting pattern across such features. A full explanation of all attributes contained in the dataset can be downloaded from:

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, Will They Blend? Amazon S3 Meets MS Blob Storage plus Excel, blends data from Amazon S3 and MS Azure Blob Storage and it is downloadable from the KNIME Community Hub.



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

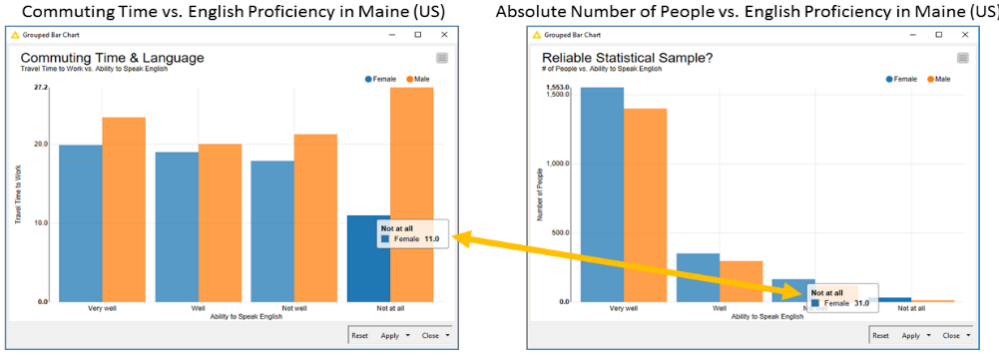
Yes, they blend!

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 are displayed in the bar plot on the left in the figure below.

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 further 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 the figure 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!



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 dataset by English fluency.

> ***Note.*** 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 dataset 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, is indeed dedicated to all such \ ridiculous conclusions that could be easily unmasked just by using some common sense.

Local Files Meet Remote Files

Will Blending Overcome the Distance?

Author: Rosaria Silipo, KNIME

Workflow on KNIME Community Hub: Will they blend? Local vs. Remote Archived File

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, browse 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. The airline data that we use in this experiment originally come from the website of the Bureau of Transportation Statistics. We focus on December 2007 and December 2008. Let's imagine that I worked on this dataset for another project and I already have the data for 2008 compressed and stored locally on my laptop. I am missing the data for 2007 but my colleague worked with these data before and made a compressed version available on her public space on the KNIME Community Hub.

So on the one hand I have a compressed CSV file with the data for December 2008 from the airline dataset here on my laptop. And on the other, I have a link to a compressed CSV file with the data for December 2007 on some server in some remote location. Will KNIME fill the distance? Will they blend?

Topic. Departure delays by carrier

Challenge. Collect airline data for December 2007 and December 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 HTTPS connection

Integrated Tool. JavaScript based visualization.

The Experiment

Access the LOCAL file with airline data for December 2008

The airline data for December 2008 have already been downloaded to my machine for a previous experiment. Data are still compressed. I only need one CSV Reader node available with new File Handling framework, updated in version 4.3 of KNIME Analytics Platform, to import the compressed CSV file directly into the KNIME workflow.

Access the REMOTE file with airline data for December 2007 via HTTP connection

Although stored remotely, the data for December 2007 can also be imported directly into a KNIME workflow without downloading it first. I just need to set up the HTTPS connection and read the data into a KNIME data table. The data are available on the KNIME Community Hub:

https://hub.knime.com/lada/spaces/Public/latest/flights_december2007.csv.gz.

With the new File Handling updated in version 4.3, new Connector nodes were made available that can connect to remote file systems. They can be found in the IO/Connectors in the Node Repository. Besides, most Reader nodes support the dynamic ports allowing to access files directly from the remote file systems.

In this case:

- We first established an HTTPS connection to the server URL (https://hub.knime.com) through an HTTP(S) Connector node,
- By activating the dynamic port in the CSV Reader node, we then imported the compressed CSV file directly from the remote location, without additional steps of downloading and decompressing.

Blend the two datasets

Now, the lower branch of the workflow deals with the airline data for December 2008 from the local file, while the upper branch handles the airline data for December 2007 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 (ORD), and calculates the ratio of departure delays by carrier, which the Bar Chart node then interactively visualizes.

This workflow, Will they blend? Local vs. Remote Archived File, is available on the KNIME Community Hub.

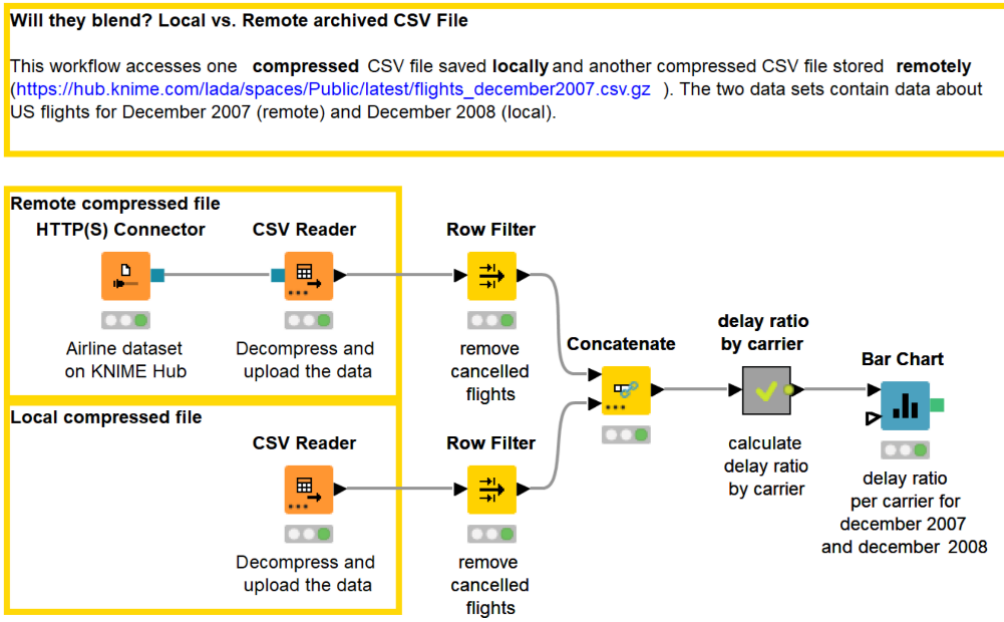
The Results

Yes, they blend!

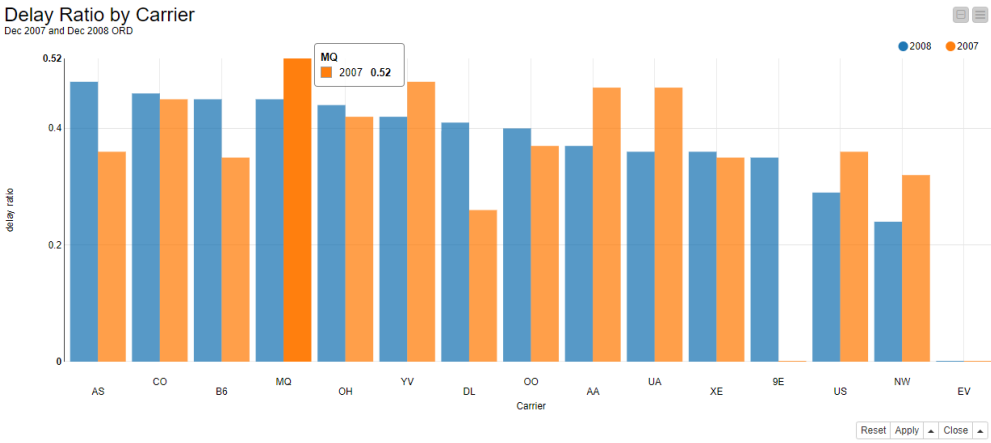
By looking at the chart we can see that if you had taken an American Eagle Airlines (MQ) flight from Chicago in December 2007 you would have been delayed at departure one out of two times. Things would have looked better though one year later in December 2008. Delta Airlines (DL) and Northwest Airlines (NW) seemed to be the most reliable airlines when departing from Chicago O'Hare airport respectively in December 2007 and December 2008. Pinnacle Airlines (9E) and Atlantic Southeast Airlines (EV) did not have flights in Chicago O'Hare airport in December 2007. For EV in December 2008 there were no delays but only 9 flights were made.

In this article 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!



This workflow successfully blends data from a local and a remote file location. Both compressed CSV files are read with CSV Reader without additional decompressing. The remote file is accessed through an HTTP connection.



Bar chart of departure delay ratio by carrier for December 2007 and December 2008.

MS Word Meets Web Crawling

Identifying the Secret Ingredient

Author: Heather Fyson, Lada Rudnitckaia, & Roland Burger, KNIME

Workflow on KNIME Community Hub: Will they blend? MS Word Meets Web Crawling

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 via drag and drop from the KNIME Community Hub.

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

Crawling the web page http://www.handletheheat.com/peanut-butter-snickerdoodles/.

1. First a Table Creator node is introduced to contain the URL of the web page.
2. Then the Webpage Retriever node is applied to retrieve the content of the web page as a string.
3. Finally, we split and ungroup the individual lines of the extracted web page, filter the ingredients list items, convert them from String to XML, and parse the ingredient names from these items via XPath node.

Now that we have the ingredients, we repeat the same steps from 2 to 5 as described above in “Extracting the list of ingredients”.

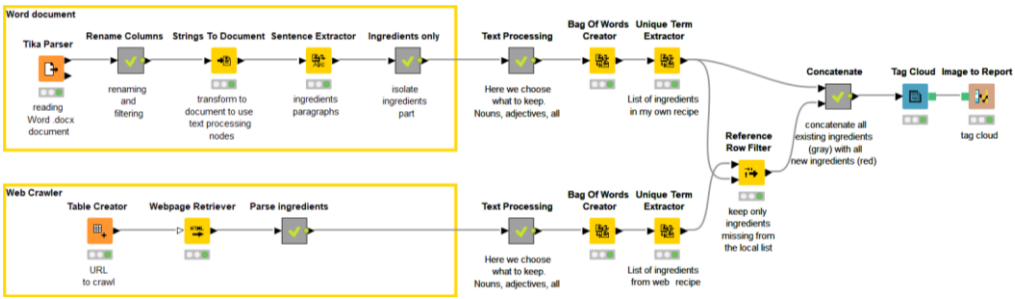
Comparing the two lists of ingredients

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 below.

The workflow used for this experiment, Will they blend? MS Word Meets Web Crawling, is available on the KNIME Community Hub.

The Results

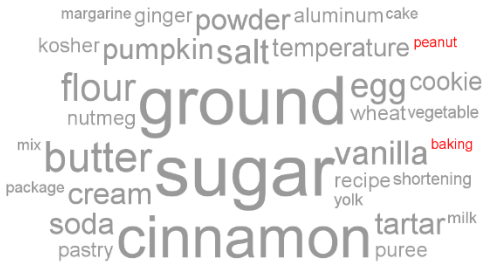


This workflow successfully blends ingredients from recipes in a Word document with ingredients from a recipe downloaded from the Web.

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 the figure below, you can see the words “peanut” and “baking” emerging red from the gray of traditional ingredients. Obviously, “baking” refers to the baking soda that is used in both recipes. However, tagging such words as “baking” can be ambiguous without the context. It was tagged as a verb in our upper part of the workflow and was successfully excluded from the list of ingredients, but it was tagged as a noun in the lower part and appeared as a unique ingredient in the final report. However, we know that the real secret ingredient is a peanut. 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!



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/

R Meets Python & KNIME

A Cross-Platform Ensemble Model

Author: Vincenzo Tursi, KNIME

Workflow on KNIME Community Hub: Will they blend? An Ensemble Model from R, Python, and KNIME Models

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

We used data from the "airline data set" (the data originally come from the website of Bureau of Transportation Statistics) 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

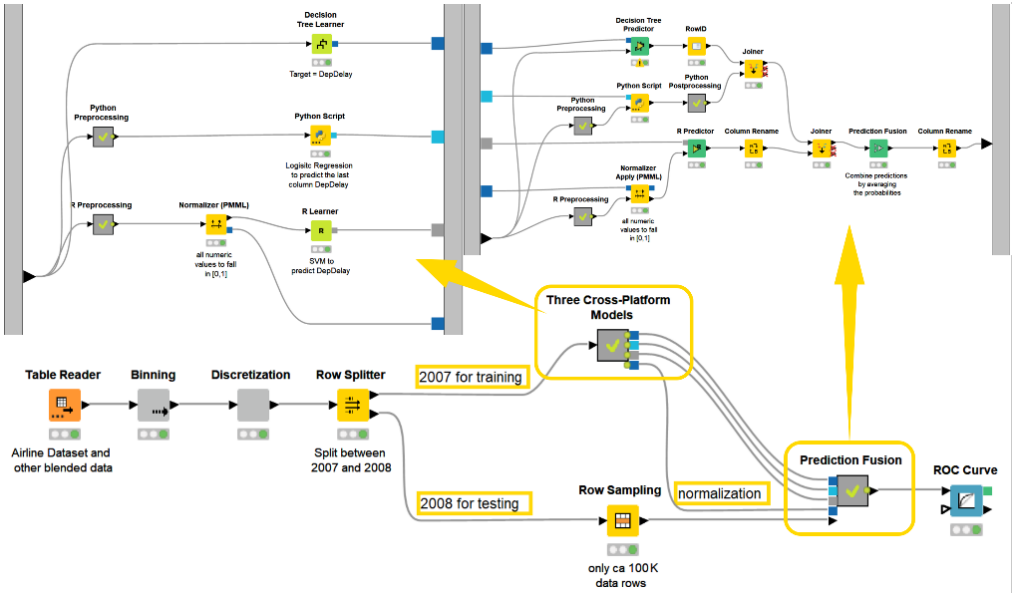
1. *Binning*. Some columns, such as distance and departure and arrival times, were binned into only a few segments.
2. *Missing Values*. Data rows with no Departure Delay value were removed. Other missing values were substituted with specific knowledge-based fixed values. Then, all string values were 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.

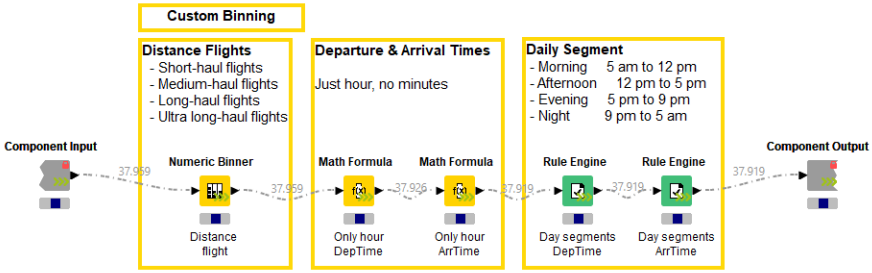
1. A Decision Tree Learner node was used to train a decision tree on KNIME Analytics Platform.
2. An R Learner node was implemented to run the R script that trains a Support Vector Machine in R. Please notice that the R svm() function has scalability issues and therefore a Row Sampling node is used to reduce the training set size to 10000 randomly extracted rows.

3. A Python Script node was introduced to execute the Python script to train a Logistic Regression model using Python scikit-learn 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 Prediction Fusion node, as it is more flexible for the integration of external models.



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.



The Streaming Executor engine executes nodes concurrently inside the component.

Note. Have a look at the first component 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

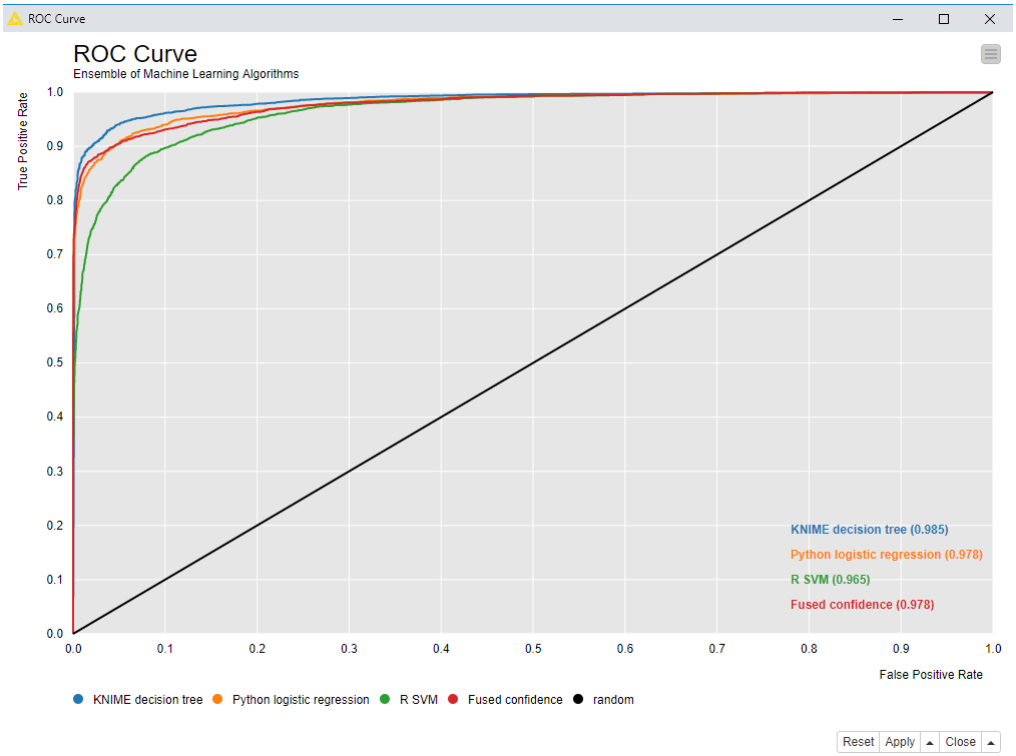
Yes, they blend!

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.

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, Will they blend? An Ensemble Model from R, Python, and KNIME Models, is available on the KNIME Community Hub.



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).

Twitter Meets PostgreSQL

More Than Idle Chat?

Author: Rosaria Silipo, KNIME

Workflow on KNIME Community Hub: Will they blend? Twitter and PostgreSQL Database

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*

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. Make sure to provide detailed and reasonable answers in the application form and note that approvement can take time. With the creation of the account, we received an API Key with API 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.
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 Configuration 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 DB Table Selector, to select table “tweets”, and two DB Row Filter nodes, to select the tweets in the archive posted between one month ago and the day before yesterday.
4. The last node, DB 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 Business Hub

When the workflow was finished, it was moved onto the KNIME Business Hub (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 DB/Connection 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 DB Connector node. This should connect to all databases, provided that you have the right JDBC driver file.

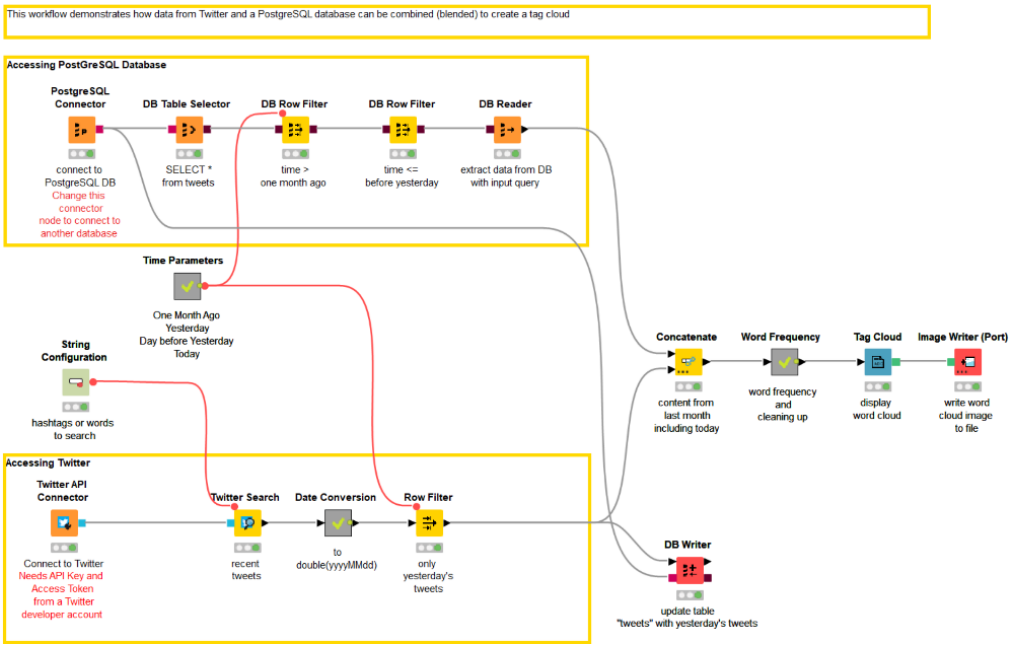
This workflow, Will they blend? Twitter and PostgreSQL Database, blends data from Twitter and PostgreSQL and builds this mini DWH application and is downloadable from the KNIME Community Hub.

The Results

Yes, they blend!

The resulting word cloud for tweets posted between October 31 and November 24, 2016, and containing the hashtag #KNIME, is shown below. Looks familiar? If you have

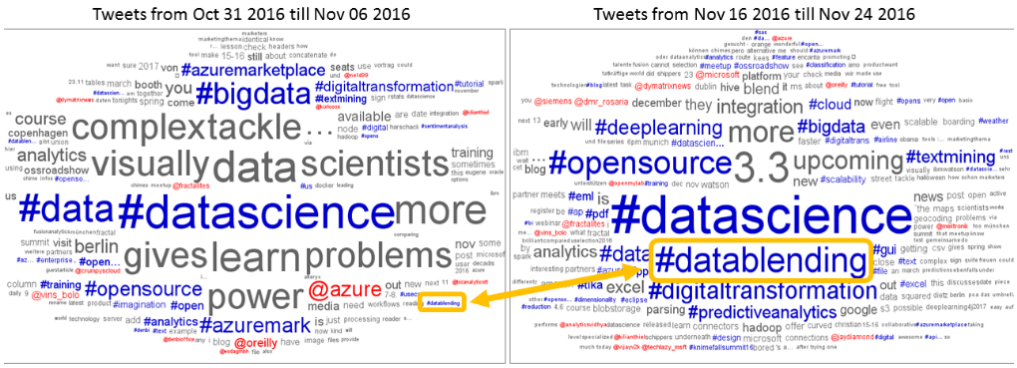
been at the KNIME Spring Summit 2016 in Berlin, this word cloud, built from the tweets containing hashtag #KNIMESummit2016, was projected during the event pauses.



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.

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.

Word cloud generated from tweets posted between Oct 31 and Nov 24, 2016 with hashtag #KNIME.



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. 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!

Hadoop Hive Meets Excel

Your Flight is Boarding Now!

Author: Vincenzo Tursi, KNIME

Workflow on KNIME Community Hub: Will They Blend? Hadoop Hive meets Excel

The Challenge

Today's challenge is weather-based - and something we've all experienced 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 the one hand we have the whole archive of US flights over the years, something in the order of millions of records, which we could save on a big data platform, such as Hadoop Hive. On the other, 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 year 2007 from the "airline data set". The "airline data set" has been made available and maintained over the years by the U.S. Department

of Transportation's Bureau of Transportation Statistics and it tracks on-time performance of US domestic flights operated by large air carriers. We could store these data on an Apache Hive platform in a cluster of the AWS (Amazon Web Services) cloud.

> ***Note.*** If we indeed had our data stored on, e.g., a Hive installation on a Cloudera cdh5.8 cluster running on Amazon cloud, we would use the dedicated Hive Connector node. The Hive Connector node, like all the 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). However, in this example we create local big data environment with the Create Local Big Data Environment node. It is very easy to use if the data is not too big for local space.

2. The workflow starts with the metanode "Read Data Into Local Big Data Env." where we first create the local big data environment. We leave the default configuration of the Create Local Big Data Environment node. Note that SQL Support is crossed for HiveQL and provide JDBC connection to support HiveQL syntax and work with a local Hive instance. Then we use a DB Table Selector node to select the airline data set table and a DB Row Filter node to extract only Chicago O'Hare (ORD) as the origin airport.

3. The goal here is to assess the dependency of flight departure delays from local weather. Daily weather data for all major locations in the US are available at https://www.ncdc.noaa.gov/cdo-web/datasets/ in the form of Excel files and can be read into KNIME Analytics Platform with the Excel Reader node. Description of the weather variables can be found here. Weather data for ORD airport only have been downloaded to be joined with the flight records extracted from the local big data environment in the previous step.

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

   Option 1. In-database join.

   4a. DB Loader node imports the weather data from Excel into local big data environment.

   4b. Thus the DB 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 transform

categories into number, to remove rows with missing values and to normalize the data.

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.

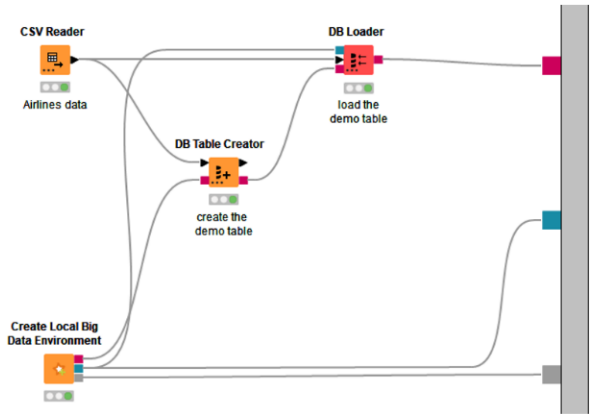
Option 2. In-KNIME join.

4a. Selected flight data is imported from the local big data environment into the KNIME Analytics Platform using a DB Reader node. The DB 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.

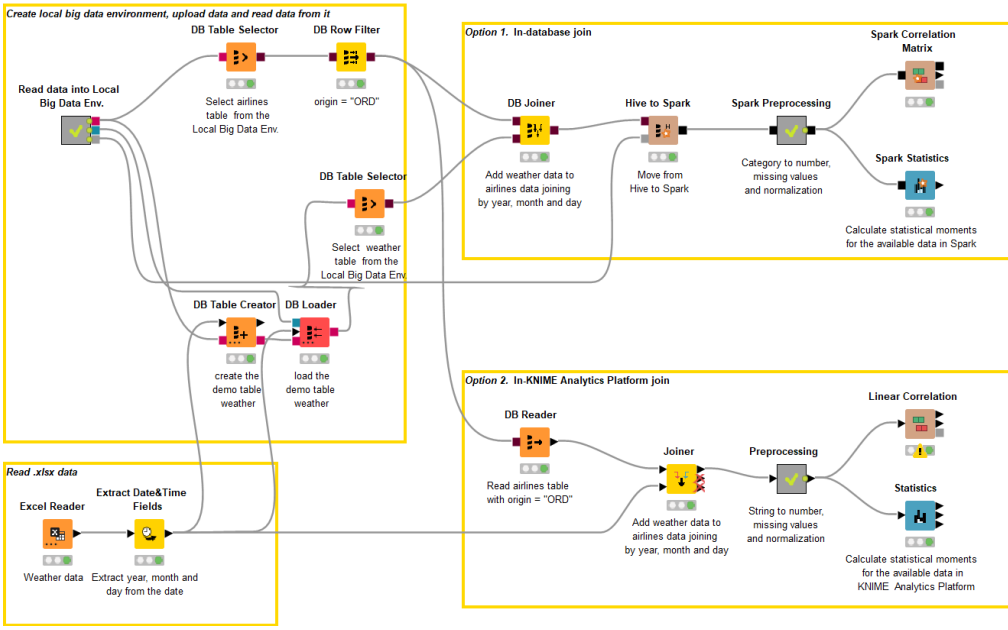
4d. Finally, the Statistics node and the Linear Correlation node calculates the same statistical measures and correlations between departure delays and weather related variables.



The "Read Data into Local Big Data Env." metanode.

The KNIME workflow, including a branch for each one of the two options, is shown 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!



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

The Results

Yes, they blend!

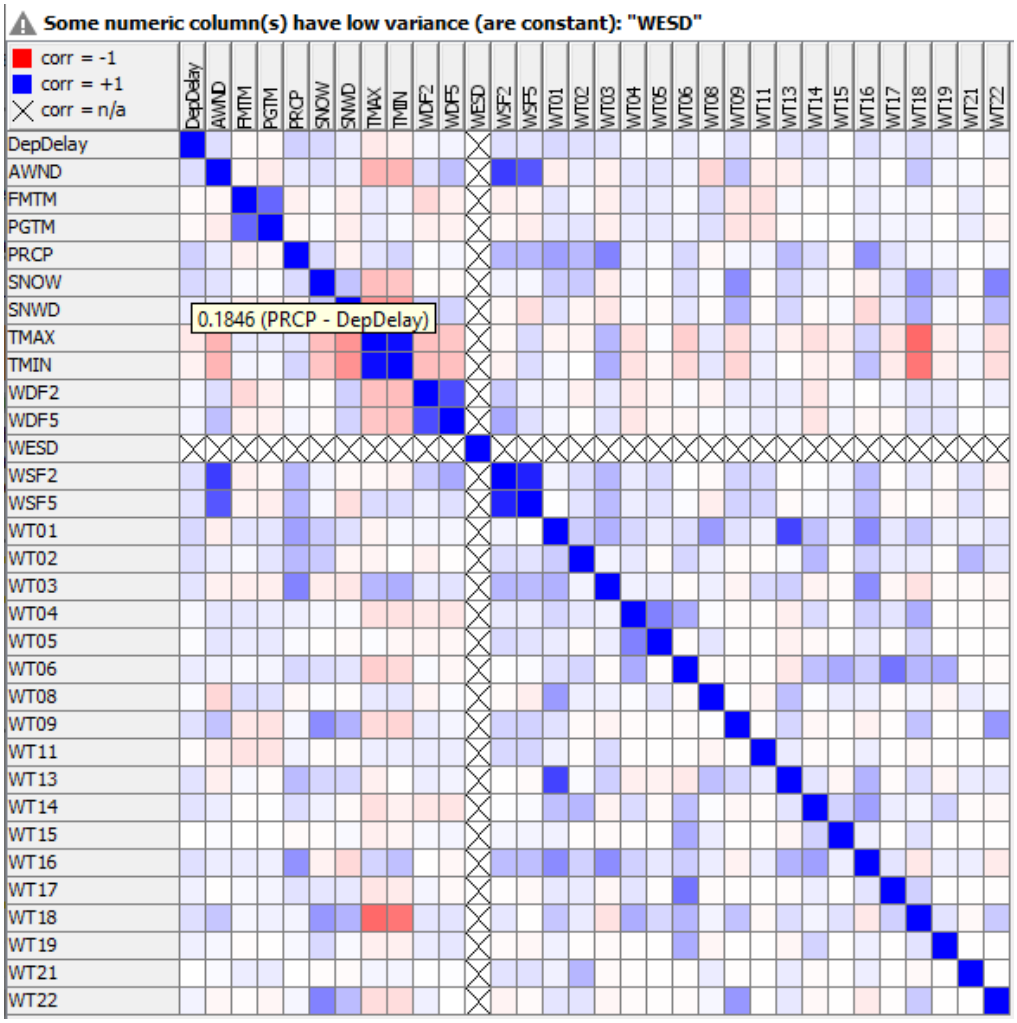
The correlation matrix shows some correlation between departure delays in Chicago and precipitation (0.18), snowfall (0.15), foggy weather (0.15) and wind speed (0.13). The correlation levels are not high since these weather conditions are not the only cause for flight departure delays. Though small, the correlation of weather variables and departure delays is hardly surprising.

However, what is surprising in this experiment is how easy it was to blend data from a modern big data environment (any really) with a traditional Excel file!

The experiment could have run on any big data platform by just changing the metanode that creates local big data environment to the generic or dedicated connector node at the beginning of the workflow. 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 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 big data environment on one side and Excel files on the other side, we can conclude that … yes, they blend!



Correlation Matrix of flight and weather data.

Open Street Maps Meets CSV Files & Google Geocoding API

Exploring the World with Geodata

Author: Rosaria Silipo, KNIME

Workflow on KNIME Community Hub: Will they blend? Text File - Google Geocode API - Open Street Maps

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 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 we want to:

- Blend the city data from the CSV file with the city geo-coordinates from the Google Geocoding API into KNIME Analytics Platform
- 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 requires the API key to authenticate each request. Google Geocoding REST API is not free but allows free trial credit for 12 months. In order to use Google Geocoding REST API, you need to navigate to the Google Cloud Console and sign in with your Google account (i.e. your gmail account), then create a project, enable Geocoding API, create API key and enable billing (the automatic billing won't start after the free trial period ends until you enable it). The request will look something like this https://maps.googleapis.com/maps/api/geocode/json?address=City,Country&key=YOUR_API_KEY

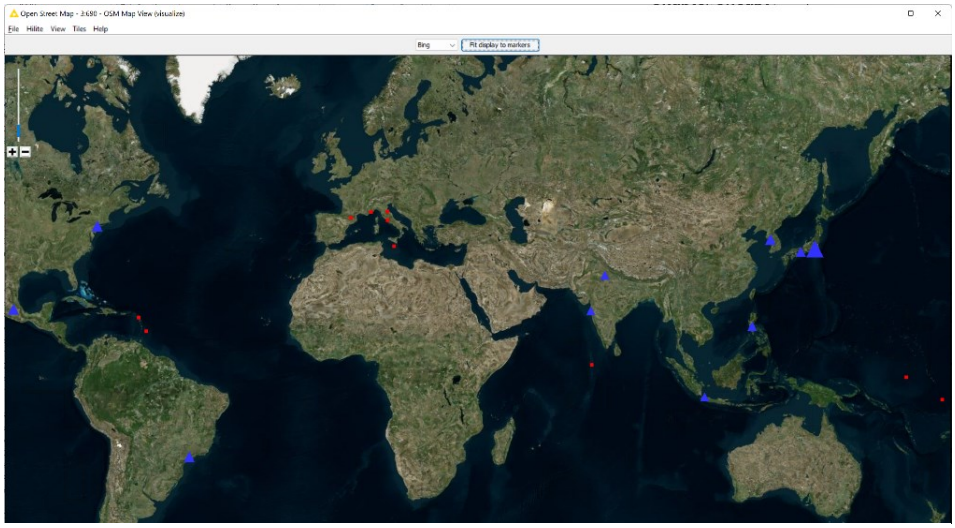
   Where:

   - *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
   - *YOUR_API_KEY* is your personal API key that you created

   The above request is saved as a template in a Constant Value Column node and appended to each city data row. Before customizing our request, we save API key as a global workflow variable. You can do that by right clicking on the

workflow in the KNIME Explorer and opening Workflow Variables window. 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. “YOUR_API_KEY” is replaced by the flow variable API Key.

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.



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

The workflow, Will they blend? Text File - Google Geocode API - Open Street Maps, is 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 below. It is available on the KNIME Community Hub.

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.

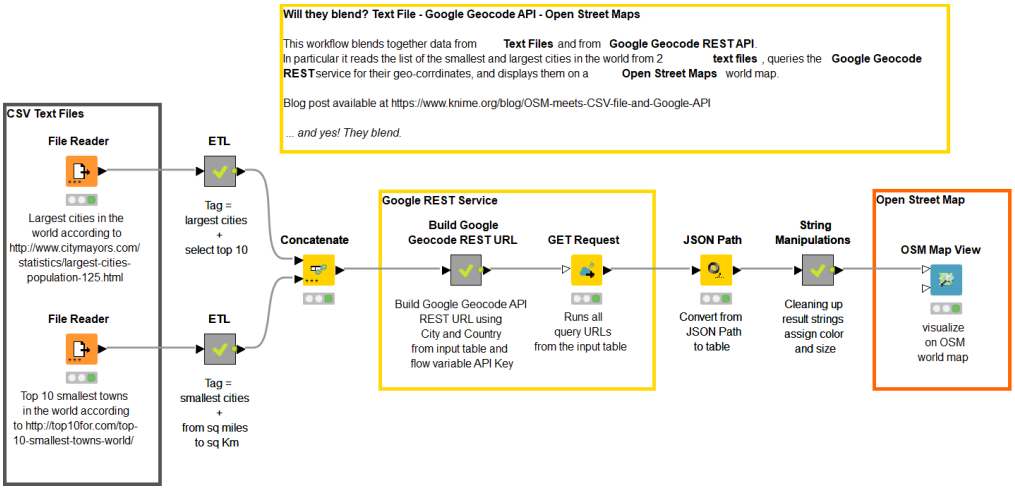
Notice the strange case of Rome, included among the largest cities but also including one of the smallest capitals, the Vatican.

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 Maps.

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

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



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.

Node and Topic Index

G

H

I

J

K

L

M

N

O

P

R

S

T

W

X

Y

Z

Will They Blend? - The Connector Collection

Data blending techniques for 50+ data sources and external tools: SQL and noSQL, Sharepoint and SAP, R and Python scripts to text and images, and more.

Dr. Rosaria Silipo has been mining data since her master's degree in 1992. She kept mining data throughout all her doctoral program, her postdoctoral program, and most of her following job positions. She has many years of experience in data analysis, reporting, business, intelligence, training, and writing. In the last few years, she has been using KNIME for all her data science work, becoming a KNIME trainer and evangelist.

Lada Rudnitckaia developed a deep interest in data analytics during her bachelor's degree program in Mathematical Methods in Economics. After gaining experience as a risk analyst, she switched to data science, pursuing her Master of Science degree at the University of Konstanz, where her focus was on machine learning in natural language processing. Currently Lada is part of the Evangelism team at KNIME and works on data science topics ranging from data access and analytics to Explainable AI.