Daimler Trucks & Buses Tech and Data Hub

Big Data Technical Challenge

Carlos Godinho

cma.godinho@gmail.com

Version 1.0, April 2019

1 Answers

Analyze 3 datasets to produce a dataset that answers the question:

Which doctors from which medical school prescribe drugs of a certain family and type?

And answer 3 proposed questions:

1 - The following table from the above 3 datasets. Please compile the data from the above 3 data sets.

Read chapter 3 - Exploratory Data Analysis for the detailed data analysis.
Check the proposal for the consolidated dataset in chapter 3.6 - Final Dataset.
A Spark application (Consolidation) is available to implement the proposal and is described in chapter 4 - Solution.

2 - In the case that the data was available as a web download (not as a downloaded file), modify the program to read from a HTTP source (S3 / Blob storage). Is it possible to process the file without local storage / staging in HDFS?

Data is available as a web download, as described in chapter 2.4 - Other Data Availability.

As available in Spark Scala doc:
https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.SparkContext
Spark reads directly from HDFS, S3 or a local filesystem. Direct reading as HTTP(S) is not supported.
A specific implementation is added to Consolidation application to parse HTTP(S) input files. The implementation is available in class Manager, method webFile. The solution is working as described in chapter 4.5 - Execution. However, this solution is not recommended for large files (> 1 million records).

An alternative solution is presented in Figure 1. Data is downloaded, parsed and an injected into Kafka. Spark Streaming read records from Kafka and processes data. Final data is stored in a final device.

Figure 1 – Alternative solution

Therefore, it is possible to process the file without local storage / HDFS.

3 - How can the original data be downloaded and deflated directly from spark? Write a spark job for this?

Spark application (Consolidation) execution and tests are described in chapter 5 - Tests.

2 Datasets

2.1 Dataset Drugs

Content: The primary data source for these data is the CMS Chronic Condition Data Warehouse (CCW), a database with 100% of Medicare enrollment and final-action Part D prescription drug event (PDE) data.
Location: https://data.cms.gov/Medicare-Part-D/Part-D-Prescriber-National-Summary-Report-Calendar/2n5w-7ghf is the base directory for Drugs.
https://download.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/PartD_Prescriber_PUF_Drug_Ntl_15.zip, holds the data.
Format: Microsoft Excel Open XML Spreadsheet (XLSX).
Data Dictionary: In file, tab `Data Dictionary`

2.2 Dataset Physicians

Content: This file contains general information about individual eligible professionals (EPs) such as demographic information and Medicare quality program participation.
Location: https://data.medicare.gov/data/Physician-Compare is the base directory for Physicians.
https://data.medicare.gov/Physician-Compare/Physician-Compare-National-Downloadable-File/mj5m-pzi6 holds the data.
NOTE: that there is also a link for the register 2015 Physicians:
http://medicare.gov/download/PhysicianCompare/2015/Refresh_Data_Archive_November_2015_1.zip this is the used link!
Format: ZIP with Comma Separator Values. Commas may exist within fields. In such case, fields are quoted.
Data Dictionary: In the page

2.3 Dataset Prescriptions

Content: The Part D Prescriber PUF is organized by National Provider Identifier (NPI) and drug name and contains information on drug utilization (claim counts and day supply) and total drug costs.
Location: http://download.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/PartD_Prescriber_PUF_NPI_DRUG_15.zip
Format: ZIP with Tab Separator Values.
Data Dictionary: https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/Prescriber_Methods.pdf

2.4 Other Data Availability

Data.Medicare.gov data is supported by the **Socrata Open Data platform (SODA)**, which provides an API to allow software developers to access it.

The description link is presentation here: https://data.medicare.gov/developers

The SODA API supports formats like JSON, XML and CSV. Each dataset has a specific endpoint to access the data.

Examples on how to us the API: https://dev.socrata.com/

For the drugs we have as example:

https://data.cms.gov/Medicare-Part-D/Part-D-Prescriber-National-Summary-Report-Calendar/2n5w-7ghf

With data end points like:

https://data.cms.gov/resource/5m6j-9i3f.json
https://data.cms.gov/resource/5m6j-9i3f.csv

3 Exploratory Data Analysis

3.1 Introduction

To make an effective exploratory data analysis to the provided data, R language is used. Besides basic R the following components are required:

- RStudio for IDE support;
- R Packages (`dplyr`, `readr` and `stringr`) to increase R productivity.

The R script is available as `DTB_Challenge_BigData/code/data_analysis.R`.

3.2 Entities

3 entities are identified:

- **Drug** – lists a set of drugs with prices and properties;
- **Physician** – Defines a set of Physicians, with specialties, training, alma-mater and other personal details;
- **Prescription** – Records of drug prescriptions made by physicians during 2015.

Each entity is available as a dataset (tabular format).

3.3 Size and Keys

Dataset size and leys are presented in Table 1.

Dataset	Size	Primary KEY
Drug	3 395 x 19	Drug Name + Drug Generic Name
Physician	2 013 843 x 43	NPI is the key, although multiple updates are available There are other identifiers with no applicability for this use case
Prescription	24 524 894 x 21	As stated in data dictionary, it is an aggregation of: `NPI + Drug Name + Drug Generic Name`

Table 1 – Dataset size and keys

3.4 Findings

In **Drug**:

- Drugs dataset has a composite key of `Drug Name + Generic name`. In fact, there are only 190 repetitions of field `Drug Name`. In a real scenario a discussion with a domain expert would be recommended;
- Considering both `Drug Name` and `Generic Name`, all fields are available.

In **Physician**:

- Using file form 2015;
- Entries related to the same Physician is repeated, due to updates in its descriptions;
- The pair `<NPI, Medical school name>` can be aggregated as it never changes.

In **Prescription**:

- Field `bene_count` has ~3 million empty records. According to the data dictionary, counts fewer than 11 are suppressed and are indicated by a blank;
- Key fields are always available;
- There are 1 208 entries of empty field `nppes_provider_first_name`, which corresponds to providers registered as institutions. This is a very low rate;
- Holds references to Prescribers (by field `npi`);
- Holds references to Drugs (by field `drug_name` and `generic_name`);
- Besides individual providers, attribute NPI includes organizational providers, nursing homes, group practices, non-physician practitioners, residential facilities, ambulatory surgery centers, and other providers.

Between entities:

- There are 336 092 prescriptions pointing to unavailable Physicians[1]. According to the data description not all the Physicians are available. Also, some Prescribers are not Physicians (although they have `npi`);
- All Drugs reported in Prescriptions are available in Drugs dataset.

[1] This decision requires a discussion with a domain expert.

3.5 Relationships

The relationships between entities are presented in Figure 2.

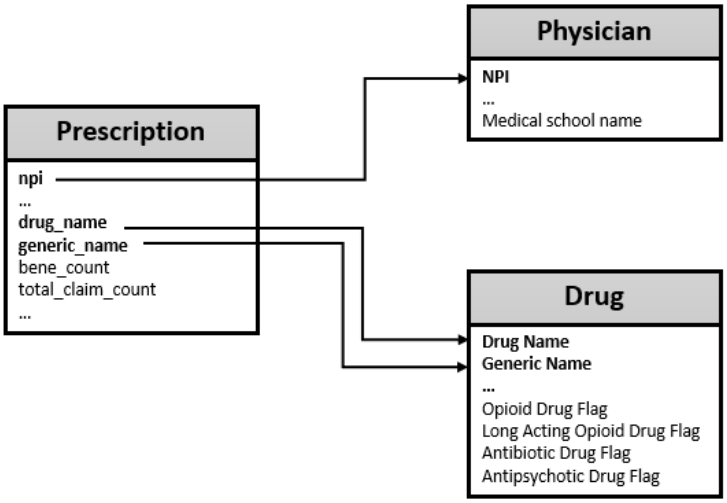

Figure 2 - Relationship between entities. Keys are represented in bold.

3.6 Final Dataset

The final dataset purposed is presented in Table 2, together with the data source location. A markdown file is available with the data dictionary as `DTB_Challenge_BigData/doc/consolidated_prescription.md`.

New Dataset		Original Entity			Notes
Name	**#**	**Entity**	**Name**	**#**	
npi	1	Physician Prescription	NPI npi	1 1	Numeric
drug_simple_name	2	Drug Prescription	Drug Name	1 8	String
drug_generic_name	3	Drug Prescription	Generic Name	2 9	String
bene_count	4	Prescription	bene_count	10	Numeric, Not always available May be useful for data scientists
total_claim_count	5	Prescription	total_claim_count	11	Numeric, May be useful for data scientists
medical_school	6	Physician	Medical school name	10	String
drug_opioid_flag	7	Drug	Opioid DF	16	Boolean Drug type descriptor
drug_long_acting_opioid_flag	8	Drug	Long Acting Opioid DF	17	Boolean Drug type descriptor
drug_antibiotic_flag	9	Drug	Antibiotic DF	18	Boolean Drug type descriptor
drug_antipsychotic_flag	10	Drug	Antipsychotic DF	19	Boolean Drug type descriptor

Table 2 - Selected fields and its origin

4 Solution

4.1 Introduction

A Spark solution validates and combines the datasets. As a result, the **Consolidation dataset** is created. To build the solution the following components are required:

- IntelliJ IDE with Scala extension;
- Scala 2.11.12
- Scala Test 3.0.5
- Spark 2.4.0
- sbt 1.2.8
- HDFS 2.9.2
- Apache HTTP server 2.2 (for simulating HTTP download)
- Red Hat Enterprise Linux Server release 6.9 (Santiago)

The project is called **Consolidation** and is available in `DTB_Challenge_BigData/code/Consolidation`.

4.2 Architecture

The following points are the result of design/architecture decisions:

- The application executes in a Spark Cluster and accesses HDFS;
- For the use underlying data structures, RDDs are used. RDDs guarantee best performance in data processing. As an alternative the SQL/Dataset/Dataframe could also be used;
- The application is available as a jar to be launched in a cluster with spark-submit;
- For flexibility, the application receives a set of arguments when launched;
- It is recommended to have input data available in HDFS, as an alternative an URL may also be provided;
- Results are provided to an output directory in HDFS;
- Input data files have fields separators, but the field separator may also be used inside a file. In this case the fields are quoted. For simplification the output data uses "," as single separator and removes quotes from fields. In the case "," are used in a field, it is replaced by ";";
- Prescriptions with field `nppes_provider_first_name` blank are removed, as they represent and organization[2];
- Scala test is used for testing a class and to show its potentiality;
- Each application execution produces a log in the output directory (`consolidation.<date+time>.log`**),** detailing its activities and timestamps;
- Data is joined by **inner join** concept, if no matching is found, entries are not considered.
- sbt script is available for producing the application.

[2] This decision requires a discussion with a domain expert.

4.3 Performance

The following performance improvements are available:

- **Kyro** serialization is activated to improve performance as data needs to be changed between executors;
- As Drugs have a small size compared to the other datasets, it is more efficient to transform Drugs data into a Map and use it as a lookup. Drugs data is Broadcasted into the executors as a name value Map;
- Filtering only relevant fields reduces the amount of data to be processed and shuffled in the cluster;
- As the application is working in a cluster, adding more resources (**horizontal scaling**) improves performance without the need of any changes in its internal implementation;
- Many physicians have its medical school reported as other, a possible performance improvement is to eliminate such elements during parsing (**not implemented**)[3];
- AS data access to HDFS is slow, an alternative is to use a distributed in memory file system. An interesting solution is Alluxio, which integrates very well with Spark **(not implemented**).

4.4 Environment

Consolidation application runs in Spark cluster of 4 servers with the details presented in Figure 3.

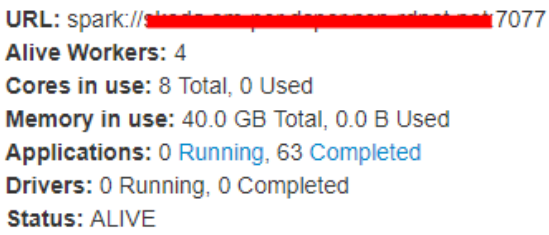

Figure 3 – Spark cluster

Each server has 2 cores, with 4 GB per core available. Total memory per server is 16 GB.

The cluster has a HADOOP cluster with 3 nodes as presented in Figure 4.

Configured Capacity:	504.48 GB
DFS Used:	11.3 GB (2.24%)
Non DFS Used:	14.53 GB
DFS Remaining:	453.01 GB (89.8%)
Block Pool Used:	11.3 GB (2.24%)
DataNodes usages% (Min/Median/Max/stdDev):	1.91% / 2.14% / 2.66% / 0.31%
Live Nodes	3 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)

[3] This decision requires a discussion with a domain expert.

Figure 4 – HDFS cluster

Figure 5 presents the recommended directory tree in HDFS.

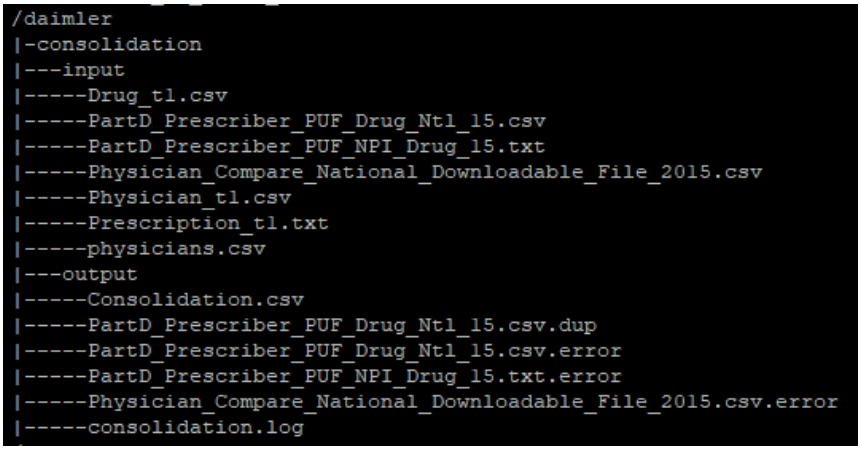

Figure 5 – Data structure in HDFS cluster

So, besides a base directory, the application expects an `input` and `output` directory.

Application jar file and launching scripts are available in NFS sharing point, accessible by all Spark nodes, as presented in Figure 6.

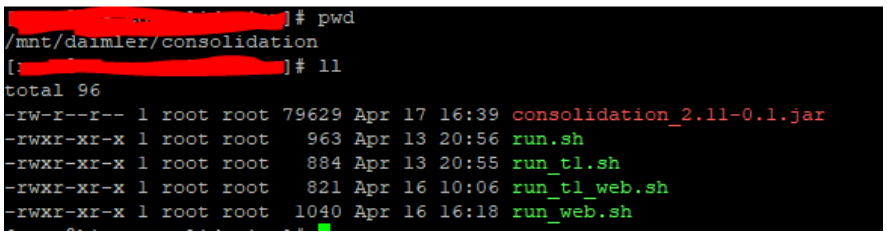

Figure 6 – Data structure in NFS

4.5 Execution

To run the application, use the following command:

```
$SPARK_HOME/bin/spark-submit --master spark://<spark address> consolidation_2.11-*.jar $1 $2 $3 $4 $5 $6
```

where:

$1 – HADOOP configuration files CSV list (`core-site.xml` and `hdfs-site.xml` are mandatory)
$2 – HADOOP base path
$3 – Drug file (must be inside HADOOP base path + input), if file in a web link, just provide an URL)
$4 – Physician file (must be inside HADOOP base path + input), if file in a web link, just provide an URL)
$5 - Prescription file (must be inside HADOOP base path + input), if file in a web link, just provide an URL)
$6 – Consolidation output file

Shell scripts are available in directory `DTB_Challenge_BigData/tests/execution`.

5 Tests

5.1 Introduction

The following test strategy is defined:

- Create a small data set to verify the consolidation business logic. Sample files are provided in `DTB_Challenge_BigData/test/test_data`. This technique allows to identify and correct issues in business logic more efficiently. Different test cases may be created by duplicating these files in a text editor;
- Setup an HTTP server to test input data provided as HTTP requests. Each dataset is available by a direct web URL;
- Copy the 3 datasets to HDFS and execute the test with the complete data.

5.2 Results

In `DTB_Challenge_BigData/test/execution` it is possible to find the shell scripts that launch the application and a set of pictures showing Consolidation application being executed in the cluster and using the its full capability.

Document `DTB_Challenge_BigData/doc/execution_times.xlsx` gathers **10 executions** of Consultation application (with full data) in the available cluster. The statistics from the execution are resumed in Table 3. The total execution time expected is **~ 13 minutes**. Each execution log is available in `DTB_Challenge_BigData/test/results/consolidation.<date-time>.log`.

Statistic	Drugs Parse, Map & Broadcast	Physician Parse	Prescription Parse	Consolidation + Write to HDFS	Total
Max	01:50	00:49	05:14	07:00	14:42
Min	01:30	00:30	03:39	06:06	12:29
Average	01:41	00:38	04:17	06:36	13:16

Table 3 – Execution statistics, in mm:ss

The final Consolidation dataset produced has a size of **20 024 410 x 10**. This means that **82%** of the initial Prescription entries are enriched with the necessary Physician and Drugs data. The gap is explained per:

- Prescriptions from institutions and not individual Physicians;
- `NPIs` reported in Prescription, but not available in the Physicians
- Repeated Drugs removed during parsing.

A sample of 1000 entries from the produces Consolidation dataset is available in `DTB_Challenge_BigData/test/results/`.

Each issue in input data is reported to a specific log file in the `output` directory as presented in Table 4. Such files are available in `DTB_Challenge_BigData/test/results/`. The name of the input file is used to generate the error file, adding an extension `error` or `dup` accordingly to the error type.

File / Source	Error Type	Physician Parse
PartD_Prescriber_PUF_Drug_Ntl_15.csv.dup	Duplicate	Key = [EYLEA,AFLIBERCEPT] x2 in lines [1394-1117] Key = [ACTIVE OB,PNV NO.66/IRON;CARBONYL/FA/DHA] x2 in lines [34-1060]
PartD_Prescriber_PUF_Drug_Ntl_15.csv.error	Format	Header, no real problem
PartD_Prescriber_PUF_NPI_Drug_15.txt.error	Format	Header, no real problem
Physician_Compare_National_Downloadable_File_2015.csv.error	Format	Header, no real problem

Table 4 – Error logs