Big Data Variety: On-Demand Data Integration

OSCAR ROMERO
A COLLABORATION WITH SERGI NADAL, ALBERTO ABELLÓ, STIJN VANSUMMEREN AND PANOS VASSILIADIS
DTIM RESEARCH GROUP (HTTP://WWW.ESSI.UPC.EDU/DTIM/)
UNIVERSITAT POLITÈCNICA DE CATALUNYA - BARCELONATECH
What is Big Data?

**VOLUME**

**Velocity**

**Variety**

**Veracity**

**Value**

**Variability**
Today, the Focus is on Variety

That Big Data is synonymous with large volumes of data is a **myth**

“Rather, it is the ability to **integrate** more sources of data than ever before — new data, old data, big data, small data, structured data, unstructured data, social media data, behavioral data, and legacy data”

The Variety Challenge

The Long Tail of Big Data

The ultimate goal is...

- **Integrate** new data sources **on-demand**,
  - Legacy Systems
  - External Data (typically, semi-structured or unstructured data)
  - Social Media and Behavioural Data Sources
- Provide the required flexibility for conducting **on-demand data analysis** techniques
  - Data preparation

**Data Analysis Democratisation**

**Automation of the data lifecycle**

**From Model-First (Load-Later) to Load-First Model-Later**
The Big Data Lifecycle

Data Sources
(in Volume and / or Velocity and / or Variety)

Intensive Data Flows (Ingestion)

Data Lake / Polystores / Dataspaces

Small Analytics(Q&R / OLAP) Big Analytics(DM/ML)

Business Strategy

Big Data Analytics

Big Data Management
On-Demand Data Integration

CHALLENGES FROM A TECHNICAL POINT OF VIEW
The Variety Challenge

There is no formal framework to characterize the Variety Challenge

However, they can easily be reduced to the Data Integration theoretical problem:

**But dealing with external data as first-class citizen**

- Extract, transform and load data from the sources
  
  **Analyse any external source; largely automate the process to facilitate incorporating new sources**

- Integrate the data in a common repository
  
  **Right-time data integration (i.e., automation is a must)**

- Query and exploit the data
  
  **Deterministic Vs. Undeterministic “queries”; Personalisation / recommendations**


The Theoretical Problem: Data Integration

- **Data Sources Layer**
  - Text (TXT)
  - Images (JPEG, PNG)
  - Videos (AVI, MP4)
  - Sounds (MP3, WAV)
  - SMS
  - Code (HTML, RDF)
- **Integration Layer**
  - MAPPINGS
  - Partial views
- **Exploitation Layer**
  - Mathematical Modeling
  - OLAP & Dashboarding
The Theoretical Problem: Data Integration

Two main research fields:
- Knowledge management (ontology-based data access, data exchange, data integration)
- Databases (federated databases, multidatabases, wrappers-mediators, data warehousing)

The constructs are the same in both approaches:
- Data source schemata
- Target schema
- Mappings (either LAV, GAV and GLAV)

Also they must decide between virtual vs. materialised data integration
The Theoretical Problem: Data Integration

To know more:

[1] AnHai Doan, Alon Y. Halevy, Zachary G. Ives:

[2] Maurizio Lenzerini:
Data Integration: A Theoretical Perspective. PODS 2002: 233-246

[3] Mary Tork Roth, Peter M. Schwarz:
Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. VLDB 1997: 266-275
A Change of Paradigm

FROM MODEL-FIRST (LOAD-LATER) TO LOAD-FIRST MODEL-LATER
BIG DATA VARIETY: ON-DEMAND DATA INTEGRATION

Model-First (Load-Later)

**Product**
- Popularity
- Top feature
- Bottom feature

**Feature**
- \( \text{Avg(sentiment)} \)

**User**
- \( \text{Avg rating} \)
- List of preferences

**Interested In**
- \( \text{Avg(sentiment)} \)
- Keen: \( \text{Avg(landing time)}/\#\text{visits} \)

**Assesses**
- Product homogenization (e.g., duplicate detection)

**Is part of**
- Product

**Sentiment Analysis (e.g., Text Mining)**
- User
- Tweet
- Date
- Location

**Log Analysis (e.g., Process Mining)**
- User
- Product
- Landing time
- Visits ts

**Web Logs (Logs)**
- User
- Product
- Landing time
- Visits ts

**USER WEB BEHAVIOUR**

**USER FEEDBACK**
- Twitter API (JSON)
  - User
  - Tweet
  - Date
  - Location

**PRODUCT INFO**
- In-house DB (PostgreSQL)
  - Product
  - Product features

**Data Mining**
Model-First Drawbacks

Fixed Target Schema

Product
- Popularity
- Top feature
- Bottom feature

Feature
- Avg(sentiment)

User
- Avg rating
- List of preferences

Interested In

- Avg (sentiment)
- Keen: Avg(landing time)/#visits

Assesses

Product homogenization
(e.g., duplicate detection)

Sentiment Analysis (e.g., Text Mining)

Log Analysis (e.g., Process Mining)

Web Logs (Logs)

USER WEB BEHAVIOUR

Twitter API (JSON)
- User
- Tweet
- Date
- Location

In-house DB (PostgreSQL)
- Product
- Product features

PRODUCT INFO

User
- Product
- Landing time
- Visits ts

High Entry Barriers

Permanent transformations

Sentiment Analysis (e.g., Text Mining)

Product homogenization
(e.g., duplicate detection)

Log Analysis (e.g., Process Mining)

Web Logs (Logs)

USER WEB BEHAVIOUR

Twitter API (JSON)
- User
- Tweet
- Date
- Location

In-house DB (PostgreSQL)
- Product
- Product features

PRODUCT INFO

User
- Product
- Landing time
- Visits ts
Load-First Model-Later

Data Lake

Product
- Popularity
- Top feature
- Bottom feature

Interested In
User
- Avg rating
- List of preferences

Data Views

Product
- Popularity
- Top feature
- Bottom feature

Interested In
User
- Avg (sentiment)
- Keen: Avg(landing time)/#visits

User
- Avg rating
- List of preferences

Product
- Popularity
- Top feature
- Bottom feature

Is part of
Feature
- Avg(sentiment)

Assesses
Data Views

Analyst 1

Twitter API (JSON)
In-house DB (PostgreSQL)
Web Logs (Logs)

User Feedback

User Web Behaviour

Analyst 2

BIG DATA VARIETY: ON-DEMAND DATA INTEGRATION
Load-First Model-Later Drawbacks

BIG DATA VARIETY: ON-DEMAND DATA INTEGRATION
From Data Swarms to Semantic Data Lakes

**Data Lake**

- **Catalog**
  - File 1
    - User
    - Tweet
    - Date
    - Location
  - File 2
    - Product
    - Product features
  - File 3
    - User
    - Product
    - Landing time
    - Visits ts

- **File 1**
  - **User Feedback**
    - Twitter API (JSON)
  - **Product Info**
    - In-house DB (PostgreSQL)
  - **User Web Behaviour**
    - Web Logs (Logs)

- **Data Views**
  - **Product**
    - Popularity
    - Top feature
    - Bottom feature
  - **User**
    - Avg rating
    - List of preferences
  - **Interested In**
    - Avg (sentiment)
    - Keen: Avg(landing time)/#visits
  - **User**
    - Avg rating
    - List of preferences

- **Analyst 1**
  - Data Mining
- **Analyst 2**
  - Data Mining

- **Feature**
  - Avg(sentiment)

- **Assesses**
  - Is part of

**BIG DATA VARIETY: ON-DEMAND DATA INTEGRATION**
From IT-Centered to User-Centered

Data Lake

Catalog

File 1
- User
- Tweet
- Date
- Location

USER FEEDBACK
Twitter API (JSON)

File 2
- Product
- Product features

PRODUCT INFO
In-house DB (PostgreSQL)

File 3
- User
- Product
- Landing time
- Visits ts

USER WEB BEHAVIOUR
Web Logs (Logs)

Product
- Popularity
- Top feature
- Bottom feature

Interested In
- Product
- Tweet
- Date
- Location

User
- Avg rating
- List of preferences

Assesses

Analyst 1
Data Mining

Analyst 2
Data Mining

Interested In
- Product
- Tweet
- Date
- Location

User
- Avg rating
- List of preferences

Is part of

Feature
- Avg(sentiment)

AUTOMATIC DATA GOVERNANCE

Data Views
On-Demand Data Integration, Really
Our Proposal

Metadata Definition
- Data Integration Architecture
- Metadata Management

Data Exploitation
- Virtual Data Integration
- Materialized Data Integration

Exploration
- Data Analyst
- Deploy

Metadata Stewardship
- View definitions
- Model
- Instantiate

Data Sources
- Structured static data sources
- DIF
- Q
Our Proposal

To know more about semantic-aware materialised data integration:


A Semantic-Aware Data Architecture

λ-architecture
Bolster: A Software Reference Architecture
Instantiation
Big Data Integration Ontology

ONTOLOGY-BASED DATA ACCESS


Ontology-Based Data Access

Virtual integration with LAV mappings
Big Data Integration Ontology

We revisit the Data Integration framework and construct an ontology as follows:

- Global level \((G)\) – Integrated view
- Source levels \((S)\) – Views on the data sources
- Mappings \((M)\) – LAV mappings between \(G\) and \(S\)

Example:

- Cross-domain queries on:
  - Monitored data on video players (lag ratio, etc.)
  - Tweets in English gathered through a feedback gathering tool
Global Level

Green: concepts

Yellow: attributes
Source Level – Exposed by Wrappers

Sources are exposed by means of wrappers

- We automatically bootstrap the attributes projected by the wrappers

\[ Q_1: \text{ID and compute the lag ratio} \]
\[ \text{db.getCollection('vod').aggregate([} \]
\[ \{ \text{project: "VoDmonitorId":true,"lagRatio": \text{divide } ["waitTime","watchTime"]} \} \]}

\[ Q_2: \text{all attributes for tweets in english.} \]

\[ Q_3: \text{association target app} \rightarrow \text{monitor, feedback gathering tool} \]

Red: Wrappers; Blue: Wrapper attributes
Mappings

A LAV mapping for a wrapper $Q$ is defined as: $M = <G,S>$ where:

- $G$ is a named graph (subgraph of $G$)
- $S$ is a set of triples of the form:
  - $<x, \text{owl:}\text{sameAs}, y>$
  - $<x, \text{rdf:type}, S:\text{Attribute}>$ and
  - $<y, \text{rdf:type}, G:\text{Feature}>$
LAV Mapping Example

Q1 S:provides { sup:InfoMonitor G:hasFeature sup:lagRatio . sup:VoDMonitor sup:generatesQoS sup:InfoMonitor . sup:VoDMonitor G:hasFeature sup:idMonitor }

q1:lagRatio owl:sameAs sup:lagRatio
q1:VoDMonitorId owl:sameAs sup:idMonitor
LAV Mappings (Q2)
LAV Mappings (Q3)
Answering Queries Using Views

EXPLORATORY QUERYING ON ON-DEMAND INTEGRATED DATA

Query Answering

Any SPARQL query on the global graph must be rewritten as a query in terms of the wrappers

Example of query over $G$:

```sparql
SELECT ?w, ?t WHERE
  ?t rdf:type sup:lagRatio
  ?y G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
FILTER ?w = "SUPERSEDE"
```
Query Answering

SELECT ?t ?p
WHERE {
    VALUES (?t ?p) {ex:teamName ex:playerName}
    ex:Player G:hasFeature ex:playerName .
    ex:Player ex:memberOf sc:SportsTeam .
    sc:SportsTeam G:hasFeature ex:teamName
}

Generated Relational Algebra Expression

\[ \Pi_{\text{ex:D1/pName}, \text{ex:D2/name}} \left( \text{ex:W1} \bowtie_{\text{ex:D1/teamId} = \text{ex:D2/id}} \text{ex:W2} \right) \]
Notions on the Query Rewriting Alg.
Start from a Terminal Feature
Navigate G from the Feature
Navigate G from the Feature

\[ \Pi_{w,t}(\rho_{Q_1}.lagRatio \rightarrow t(Q_1)) \]
Navigate $G$ from the Feature

```
SELECT ?w, ?t WHERE
?t rdf:type sup:lagRatio
?x G:hasFeature ?t
?x rdf:type sup:InfoMonitor
?y sup:generatesQoS ?x
?y rdf:type sup:VoDMonitor
?z sup:hasMonitor ?y
?z rdf:type sc:SoftwareApp
?z G:hasFeature ?w
FILTER ?w = "SUPERSEDE"
```
Explore Join Candidates

Any other wrapper map to these features? Yes! Q3 contains idMonitor too!
Join to an Alternative Wrapper
Continue Navigating G
Computational Complexity

Based on the theoretical data integration framework, our approach is reduced to answer queries using views (each wrapper is a view)

- Our algorithm can be reduced to the set cover problem (known to be NP-complete)

Specifically, the query rewriting algorithm is:

- Linear in the size of the subgraph of $G$ to navigate
- Linear in the size of the wrappers mappings
- Exponential in the number of wrappers that may join

However:

- Our experiments show that typically Big Data sources have few join points and therefore this exponential complexity is affordable in the real cases we tested
- Use SLAs to place heuristics to reduce the search space

Conclusions

Our approach facilitates deploying virtual integration with LAV

- It uses the same model to represent the local sources, the mappings and the global schema
- We may further exploit additional semantic annotations (e.g., implicit aggregations)
- It supports evolution (e.g., new API releases) without potential crashes
- Creating the source ontologies is reduced to define tuple-generating dependencies (\(tgds\))
  - Production rules between models (XML, JSON, CSV, Relational to RDF(S))
- Creating the global ontology can be achieved by means of entity-resolution state-of-the-art techniques on the source ontologies automatically generated
  - Pay-as-you-go integration model

As result, our system allows ad-hoc exploration on on-demand integrated data to gain insight on the data lake available sources

Oscar Romero

✉️ oromero@essi.upc.edu

@romero_m_oscar

http://www.essi.upc.edu/dtim/