Towards Mega-Modeling: A Walk through Data Analysis Experiences

Stefano Ceri
DEIB – Politecnico di Milano
stefano.ceri@polimi.it

Emanuele Della Valle
DEIB – Politecnico di Milano
emanuele.dellavalle@polimi.it

Johann-Christoph Freytag
DBIS - Humboldt-Universität zu Berlin
freytag@informatik.hu-berlin.de

Themis Palpanas
DISI – University of Trento
themis@disi.unitn.eu

Dino Pedreschi
KDD Lab - Università di Pisa
pedre@di.unipi.it

Roberto Trasarti
KDD Lab – ISTI,CNR Pisa
roberto.trasarti@isti.cnr.it

1. INTRODUCTION

Big data is perceived as a fundamental ingredient for fostering the progress of science in a variety of disciplines. However, we believe that the current ICT solutions are not adequate for this challenge. Abstractions and languages for big data management are tailored to vertical domains and influenced by underlying ICT platforms, hence unsuitable for supporting "computational interdisciplinarity", as it is required if one wants to use the best of, e.g., analytical, inductive, and simulation techniques, all at work on the same data. In other words, "our society is data-rich, but it lacks the conceptual tools to handle it" [1].

In previous work [2], we advocate the need for a new approach to data analysis, based on mega-modeling as a new holistic data and model management system for the acquisition, composition, integration, management, querying and mining of data and models, capable of mastering the co-evolution of data and models and of supporting the creation of what-if analyses, predictive analytics and scenario explorations.

In this paper, we provide some evidence that mega-modeling is a viable approach to data analysis by using a bottom-up, inductive method. We consider several experiences of data analysis research performed at our home institutions and examine them in retrospective, inducing their mega-modularization a-posteriori. This exercise convinces us that the mega-modeling approach could be highly beneficial.

2. MEGA-MODELING

In this section, we provide a historical perspective on the development of the mega-modeling concept and a self-contained summary of our previous work on Mega-modeling [2].

2.1 Unifying Data-centric Disciplines

Big Data analytics calls for *a comprehensive theory and technology that blend simulation, analytical, ontological and data-driven models into one picture*. Modeling, as we know it today, is required to scale up to a higher level, that we call **mega-modeling** [2].

The database/data mining community has been investigating approaches that unify data analysis and mining, since the seminal paper on *inductive databases* and *data mining as a querying process* by Imielinski and Mannila [11]. A fundamental aspect of [11] is the representation of data mining activities through patterns, whereas patterns can be seamlessly integrated with data and can therefore be the subject of queries; such view attempts a conceptualization and generalization of data mining. Yet, this idea has found concrete realizations only lately and partially; e.g., it is used in [12], where extracted data patterns are defined as views on top of data tables, and as such can be composed with domain-specific data representing spatio-temporal trajectories expressing human mobility [12].

The roots of mega-modeling can be traced to an article appeared in 1992 on "Mega-programming" [13] – *large, autonomous computing systems whose interfaces are described through a data-centric approach and whose execution behaviour can be inspected* – and to another one that appeared about one decade later on "mega-models" [14] – models of which at least some elements represent and/or refer to other models or meta-models – that paved the road to the school on Model-Driven Engineering (MDE) [15].

However, the innovative aspects of mega-modeling go beyond classical model-driven software generation. Mega-modeling aims at defining a comprehensive theory and technology of model construction (with an emphasis on incremental bottom-up approaches), model search, model fitness evaluation, model composition, model reuse and model evolution. We need entirely new *models of models*, namely algebras of objects representing patterns, rules, laws, equations, etc., which are either mined/induced from data, or based on deep mathematical findings or agent-based reasoning – an overarching algebra of data and models that allow us to devise a new holistic system for

integrated data and model acquisition, integration, querying and mining, capable of mastering the complexity of the knowledge discovery process.

2.2 Mega-Modules

The building block of mega-modeling is the **mega-module** – a software component capable of processing "big data" for analytical purposes. Every mega-module performs a well identified computation, which can be considered a unitary transformation from inputs to outputs. Inputs and outputs take the form of data and of patterns, where data are domain-specific both in terms of their schema end instances, while patterns are forms of data regularity or rules whose schema is domain-independent and whose content typically reflects collective or aggregated data properties; patterns may be extracted by data analysis algorithms, which may in turn be embodied within mega modules. Every mega-module can internally use data and patterns that are considered as invariant in the context of the computation, whose extension can be either local (e.g., organization-specific) or global (e.g., stored in public databases or ontologies).

Every mega-module exhibits a format that consists of three distinct phases of information processing, although such phases can vary significantly for their internal organization: data preparation, analysis, and evaluation.

- The first phase, **data preparation**, consists of the processing of input data and patterns for the purpose of assembling input objects that will be the subjects of the analysis. The distinction between data and object is of semantic nature: data preparation typically assembles several elementary data in the input to generate a single object for the purpose of analysis. The aggregative process that builds a object can be driven by a variety of purposes – abstracting irrelevant differences, recognizing common features, aggregating over elementary items which satisfy given predicates – thus, semantically interpreting and reconstructing data. The keywords for the preparation phase are: data sensing, acquisition, integration, transformation, semantic enrichment.
- The second phase, **data analysis**, consists of extracting computed objects from input data, possibly using the input patterns as references. Data analysis produces the response to a specific problem by performing the core scientific processing, and uses a variety of methods, ranging from mathematical to statistical models, from data mining to machine learning, from simulation to prediction, including crowd-sourcing as a way for asking social responses. The keywords for the analysis phase are: mining, learning, modeling, simulation, forecast.
- The third phase, **data evaluation**, consists of preparing the output objects, which may in turn be presented as data and/or patterns. This phase consists of filtering or ranking computed objects based on their relevance, and possibly of a post-processing so as to observe the result in the most suitable way for the mega-module enclosing environment or user. The keywords for the evaluation phase are: quality assessment, filtering, significance measurements, presentation, delivery, visualization.

In Fig. 1, we propose a mega-module graphical element, which visually captures the characteristics of a mega-module as described above.

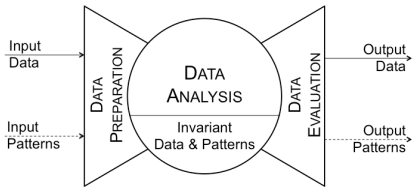


Figure 1: Visual presentation of a generic mega-module.

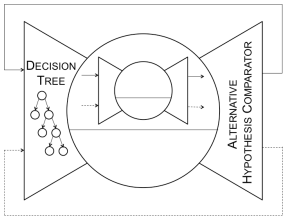
The presence of the three phases allows us to define two standard **inspection points** within a mega-module, used for asynchronous control and feedback that mega-modules should provide to their enclosing environment. The first one, after preparation, provides a view on objects abstracted/reconstructed from data; the second one, after analysis, provides a view of the objects resulting from the analysis.

A mega-module inspection consists in extracting its controls asynchronously, during its execution; this in turn allows the enclosing environment to trace mega-module execution, to estimate completion time, and to anticipate the quality of its results. We regard the data and patterns that may be exchanged by a mega-module during its execution as the **mega-module controls**. A mega-module should expose commands to the enclosing environment that may alter its behavior, for instance by rising or by lowering confidence levels during analysis based on the quality of intermediate results or on the expected completion time. It should also be possible to suspend, resume, and terminate the mega-module computation.

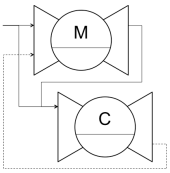
Wrapping up, we associate to each mega-module the potential of expressing classes of computations on top of big data, thereby highlighting the computational nature of the modules and the support of dynamic aspects related to inspection, adaptation, and integration. In the design or reverse engineering of Mega-Modules, data come first: clarifying their input and output data by using known pattern types is the key aspect for guaranteeing module interoperability and reuse. Emphasizing the role of data transformers for Mega-Modules opens up to using/inventing algebraic languages for data-driven orchestrations and optimizations. Moreover, the possibility of declaring the streaming or ordered nature of data opens up possibilities for a different class of optimizations that emphasize recent and ordered data.

2.3 Mega-modularizing Big Data Analysis

In mega-modeling, a big data analysis problem is modeled as a data-driven workflow involving several Mega-Modules. Composition abstractions are the means of combining mega-modules to the purpose of creating sophisticated analytical processes. Composition abstractions reflect the classical ways of assembling modules into higher order computations. Every abstraction induces a hierarchical decomposition, singling out an enclosing mega-module and one or more enclosed mega-modules; our goal is to describe computations over big data as top-down recursive applications of a well-designed collection of abstractions. In [2], we present an initial set of abstractions; they are orthogonal, but most likely incomplete, and further investigation is needed to consolidate them. The set includes traditional pipeline and parallel composition, but also typical data mining ones such as what-if control, drift control and component-based graph decomposition.

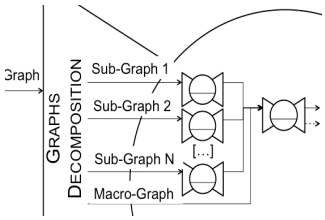


What-if control is a classical way of mining big data by exploring many alternative solutions that would occur for different choices of initial setting of models and/or parameters. Essentially, this control abstraction is a form of iteration driven by an analytical goal, allowing to repeat a mega-module under different parameterizations of input data and patterns, until a final analytical result is obtained, which possesses a desired level of, e.g., quality, precision or statistical significance; the preparation phase can be modeled by a decision tree. Many possible instances of this "what-if" iteration control may be envisage, pertaining to many existing alternatives for exploring a space of patterns/models studied, e.g., in machine learning, data mining, statistical physics and (agent-based) simulation.



Many mega-module computations are based upon the validity of underlying assumptions. Thus, if the assumptions cease to be valid, the mega-module itself must be invalidated, and then either corrected or abandoned. For instance, a credit risk predictor used by a bank for granting mortgages may become obsolete as an effect of an economic crisis that impact household incomes. The phenomenon of "drifting" describes the progressive invalidation of assumptions under which a model has been learned from data. A mega-module M, which is potentially subject to drifting, should be paired to an associated *drift-control* mega-module C, which assumes the output of M as input. The controller normally has no output, however if it perceives that the drift has occurred, then it interacts with M, by providing suitable controls.

Many mega-module computations apply to input data representing (large) networks and graphs thus making parallelization more difficult; if instead a graph has modular structure of components (namely sub-networks) with high intra-module connectivity and relatively low inter-module connectivity, then a natural parallelization can be achieved by mapping each sub-network to an internal mega-module before integrating the results using one additional combining mega-module. Such an approach enables a *component-based graph decomposition*.



Data-driven mega-modularization should also facilitate dynamic adaptation, performed by the invoking environment in the context of a mega-module orchestration. The presence of mega-module inspection points allows for asynchronously extracting parameters describing data analysis execution, while the execution is ongoing; in this way, the enclosing environment might include a **Mega-Module controller** which traces Mega-Module execution, estimates the completion time of data analysis, and anticipates the quality of its results. The controller may adaptively alter the behavior of a module, for instance by raising or by lowering the confidence levels that control the output production.

3. DATA ANALYSIS PROBLEMS

We reviewed seven recent research experiences, three in the mobility data context, and four in the data stream context.

- Problem P1: INDIVIDUAL PROFILING. Given spatio-temporal information, reconstruct trajectories and find trajectory clusters that correspond to routine daily commutes of individual citizens [3].
- Problem P2: COLLECTIVE PROFILING. Given spatio-temporal information, reconstruct trajectories and aggregate them to find typical traffic routes, each of them consisting of sequences of regions [4].
- Problem P3: REGION IDENTIFICATION. Given spatio-temporal information, reconstruct trajectories and map them to edges among cells of a spatial tessellation, then partition the resulting network of cells so as to recognize regions with high connectivity [5].
- Problem P4: TRACKING OF CROWD MOVES. Given streams of geo-tagged micro-posts (e.g., geo-tagged tweets, foursquare check-ins) from a geographic area, detect where crowds are assembling and show how they are moving using a stream of heatmaps [6].
- Problem P5: BURST OF INTEREST DETECTION. Given streams of micro-posts (e.g., tweets, facebook status), enrich them with semantic entities they talk about, before detecting bursts of interest w.r.t. the described entities [7].
- Problem P6: SENTIMENT SHIFTING. Given streams of micro-posts, extract topic(s) and the sentiment

relative to the topic(s) and monitor topic-sentiment pairs to detect sentiment shifts [8].

- Problem P7: DATA OUTLIER DETECTION. Given streams of data representing measures, summarize them by their probability density functions and then detect different kinds of outliers [9].

4. MOBILITY USE CASES

The common aspects of problems P1-P3 is the presence of trajectories as fundamental data pattern, and of trajectory reconstruction and clustering as fundamental computational steps.

4.1 Common Pattern Types

Our approach to the modeling of data analysis problems starts with the definition of the **pattern types,** i.e. generic types used for representing mobility [2]. Considering the three mobility problems, they all are based on a big dataset of observations of mobile objects, associated to their positions in space and time. Observations are assembled into trajectories that are sequences of observations of the same object, further characterized by the length and average speed. Given a set of trajectories, a medoid is the result of a statistical process that defines their median trajectory and variance. The corresponding pattern types are shown below using a simple formalism, where square brackets denote tuples, curly brackets denote sets, and the "<" ">" symbols denote lists.

Observation: [oid, position[latitude, longitude], time]
Trajectory: [tid, <Observation>, length, avgSpeed]
Medoid: [oid, <Observation>, Variance]

In addition to moving points, mobile applications also describe geographic regions, typically characterized by their geometry, which is a sequence of positions describing the region's border. Regions are typically related to each other in a network, which is a collection of nodes and arcs, where nodes are associated to regions and arcs connect two regions and are further characterized by a weight. The corresponding pattern types are below; note that pattern type denote minimal information and can be extended in each different application, e.g. regions may have a name and additional properties such as size and population.

Region: [rid, <position[latitude, longitude]>]
Network: [nid, {Region},{[Region, Region, weight]}]

All the problems essentially deal with trajectory assembling and managing. Similar trajectories can be grouped into clusters, and trajectories can be aggregated in space so as to connect regions rather than individual positions; such trajectories may be further characterized by their minimum and maximum time.

T-Cluster: [cid, {Trajectory}]
T- Pattern: [tid, <Region>, min-time, max-time>]

4.2 Models of Mobility Cases

We considered the three applications and observed that they share many aspects that can be modeled as chains of Mega-Module applications that progressively produce the relevant data.

4.2.1 Individual Profiling

Problem P1 is concerned with aggregating the trajectories of a single individual to capture her usual commutes; thus it requires combining Mega-Modules for reconstructing trajectories from observations, then to cluster the trajectories of each individual users, and then compute the medoid of each cluster and associate it with labels in order to extract opportunities for car-pooling. This chain of transformations is described in Fig. 2, which visually shows trajectory reconstruction, clustering, and profiling.

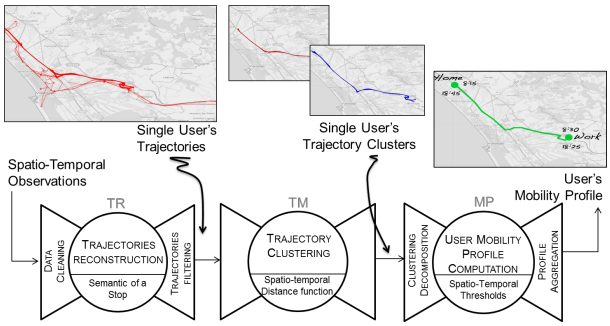


Figure 2: Problem P1

We next describe the Mega-Modules:

- **TRAJECTORY RECONSTRUCTION** (TR) – builds trajectories from observation of moving points, using a semantic description of a "stop". In the pre-processing, spatio-temporal observations received as input are cleaned, and the history of each user's movement is obtained by ordering their observations according to their time. At this point, the data analysis component processes each history by dividing it into several sub-sequences representing trajectories; each trajectory represents an individual user's trip. Data analysis is parametric and uses as input the definition of the "semantics of a stop", defined as conjunction of two spatio-temporal constraints: a minimum time span and a maximum distance between two consecutive points. Finally, the post-processing consists of filtering those trips that are not meaningful or contain outlier and anomalies that can be detected only at this level of abstraction, e.g., one-point or out-of-region trips.
- **TRAJECTORY CLUSTERING** (TM) – performs a density-based clustering of trajectory data. The data analysis is parametric, it uses as input a spatio-temporal function for computing distances between different trajectories. The result is a set of clusters of homogeneous (i.e., similar) trajectories. Trajectories may be associated with application-specific labels concerning their initial and final observations; in the example, we obtain two

groups of trajectories, tone moving from north to south and the other one moving from south to north.

- **MOBILITY PROFILE** (MP) – computes the medoids of each cluster. The pre-processing is used to partition the analysis process into separate threads and to filter those sets whose cardinality is below a given threshold. At this point, the data analysis component extracts the medoids according to a parametric distance function; each medoid is associated with a variance describing its statistical representativeness. Post-processing gathers all the results from the different execution threads, filters low-quality medoids and then constructs user's mobility profiles. In Fig. 2, two profiles are associated with a given user: (Home) 8:15 → (Work) 8:30 and (Work) 18:25 → (Work) 18:45, where "Home" and "Work" are two labels assigned to the points where their medoid trips begin and end, considering the hour of the day in which the trips occurs.

4.2.2 Collective Profiling

Problem P2 is concerned with understanding, in a broad sense, how people move across small-scale regions, thus building "important" trajectories that are relevant for further analysis (e.g. real-time traffic monitoring getting to specific locations). The first two steps for solving this problem turn out to be identical to the previous use case. However, the third step is quite different, as it involves a transformation of the most relevant trajectories from sequences of positions to sequences of regions.

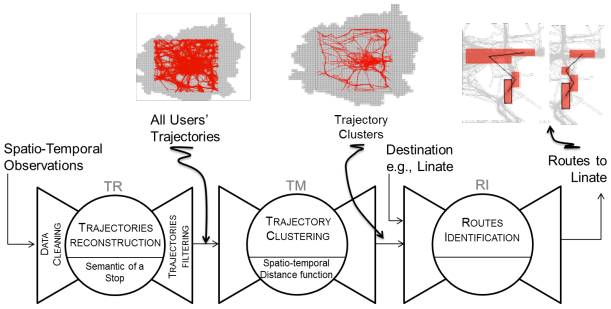


Figure 3: Problem P2

Thus, the second problem reuses the modules TR and TC and adds to them the Route Identification Mega-Module.

- **ROUTE IDENTIFICATION** (RI) - uses the trajectory clusters for mining the typical routes, represented as T-patterns. The pre-processing filters the clusters by keeping only those containing trajectories ending in a specific place specified by the application, e.g. Linate airport. Then, the data analysis component applies the T-Pattern discovery algorithm over each set of trajectories and extracts typical routes.

In the example illustrated in Fig. 3, two T-Patterns are found: the first one (on the left) represents the people arriving at Linate from the south exiting the highway at "Via Mecenate" and turning into "Viale Forlanini", while the second (on the right) represents the people arriving at Linate existing directly in "Viale Forlanini".

The former is often a smarter choice as the "Viale Forlanini" exit is often congested.

4.2.3 Region Identification

The third problem focuses on recognizing macro-regions consisting of regions that are strongly connected, i.e. such that most of traffic occurs inside the macro-region, while

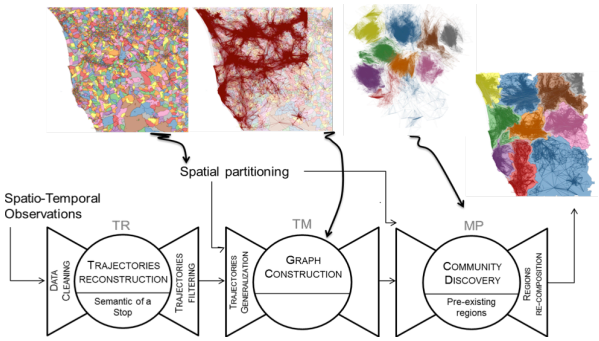


Figure 4: Problem P3

only a small amount of traffic moves between them. This problem also uses the Mega-Module for trajectory reconstruction, but it then characterizes trajectories as T-patterns traversing regions, and then describe the induced network of region connections in order to discover the macro-regions (see Fig. 4). Two new Mega-Modules are introduced:

- **GRAPH CONSTRUCTION** (GC) – transforms the trajectories into a set of sequences using the spatial tessellation as input. This is done by intersecting all the points of the trajectories with the cells and removing all the consecutive repetitions obtained. Thus, a trajectory is represented as the sequence of traversed cells without the temporal component. Then, each cell is mapped into a distinct node of the graph, and trajectories connecting two cells are mapped to edges; the weight of each edge is proportional to the number of trajectories.
- **COMMUNITY DISCOVERY** (CD) – uses Infomap algorithm, which is based on a combination of information-theoretic techniques and random walks. The result is a set of communities, each represented by a set of nodes. These sets are then remapped to the spatial dimension and joined obtaining the spatial borders of the communities.

5. TEXT STREAM USE CASES

Problems P4, P5, and P6 are based on independent experiences of the authors on text stream analytics. By studying them, we realized that they could be effectively modeled by an initial common sequence of Mega-Modules.

5.1 Common Pattern Types

The text items constituting the datasets consist of short texts associated with their authors and publication times, potentially containing a set of tags, and possibly annotated with a geo-position. Each short text is enriched by a set of topics – e.g. hashtags or semantic entities extracted from a KBs - and then annotated with the author's sentiment about each topic. The corresponding pattern types are shown below.

ShortText: [sid, {word}, user, {tag}, position[latitude, longitude},time]
Topic: [tid, {word}]
EnrichedShortText: [eid, ShortText, {Topic, Sentiment}]

Short texts are produced, enriched and analyzed on the fly. They are managed as continuous flows of textual information, i.e., text streams. The pattern types for streams of texts (of both cases) are timestamped sequences:

TextStream: [stid, <ShortText,timestamp>] or
[stid, <EnrichedShortText,timestamp>]

Further analysis of these text streams may require the introduction of other pattern types: heatmap streams, which carry the aggregated information about geographic areas from where most of the short text come from, topic popularity streams, which carry the aggregated information about the most trendy topic under discussion, and contradictions, which are the time points when positive and negative sentiments have been simultaneously recorded with respect to a specific topic, or the time points when a sentiment shift has occurred (i.e., positive sentiments on a specific topic have turned into negative, or vice versa).

HeatmapStream: [stid, <Region, value, timestamp>]
TopicPopularityStreams: [stid, <Topic,value,timestamp>]
Contradiction Stream: [stid, <topic, time, timestamp>]

5.2 Models of Text Stream Cases

In casting problems P4, P5 and P6 as a mega-modeling process, we identified a pipeline for short text enrichment that is common to all three problems.

5.2.1 Short Text Enrichment Pipeline

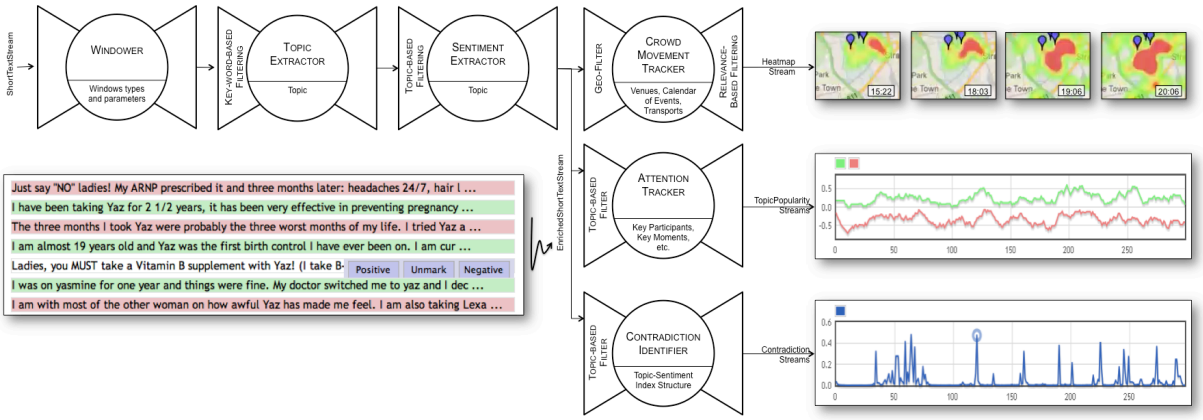


Figure 5: Problem P4, P5 and P6.

In the pipeline, the (infinite) incoming text stream is chunked in manageable blocks of short texts using windows, then each short text is parsed in order to detect the topics that are mentioned, and subsequently, the sentiment expressed in the short text for each one of these topics is extracted. Three new Mega-Modules are introduced:

- **Windowing** (W) – transforms a portion of a (by definition infinite) stream in a (finite) window – a block of processable information. Several types of windows exist, the most frequently used are: physical windows, which can hold a fixed number of data items, logical windows, which contain all the data elements received in a given time period, tumbling windows, whose content does not overlap, and sliding windows whose content overlaps. We use physical tumbling windows.
- **Topic Extractor** (TE) – extracts the topic, or topics that are mentioned in a short text. Evidently, when the short text is large enough, more than one topic may be mentioned. The topic extraction Mega-Module may look in the short text for a pre-defined set of topics (e.g., a list of topics that has been constructed based on domain knowledge, or a preceding processing step), or it may discover ad-hoc topics.
- **Sentiment Extractor** (SE) – extracts the sentiment corresponding to each of the topics mentioned in the working block. The sentiment value expresses the opinion represented as a discrete (e.g., negative/neutral/positive) or continuous (e.g., in the interval [-1,1]) value, of the subject towards a specific topic. In Fig. 5, sentiments are represented as real values between -1 and 1.

5.2.2 Text Stream Analytics Mega-Modules

The blocks of enriched short texts delivered by the reusable pipeline can then be further processed. In modeling our past text analytics experiences, three new Mega-Modules are introduced:

- **CROWD MOVEMENT TRACKER** (CMT) – tracks the movement of the crowds using geo-tagged user statements, and produces a time series of heat maps. The analysis can be limited to a given area (e.g., in [6] the London Olympic stadium and the train and metro stations around it, in [7] a tourist district of Seoul) and can be aware of the position (and shape) of the venues, the calendar of the events, and other background information. For instance, the series of heat maps in the upper-right corner of Fig. 5 show a crowd entering the Olympic stadium for the London 2012 Olympic Games opening ceremony.
- **ATTENTION TRACKER** (AT) – tracks the attention of the crowds using sentiment about a topic in the enriched short text and outputs a topic popularity stream.
- **CONTRADICTION IDENTIFIER** (CI) – organizes the information in the enriched short texts in an (incrementally maintainable) index structure that is used for efficiently managing the information on the sentiments expressed on various topics over time. This index structure is then used for identifying the time intervals and topics for which a sentiment-based contradiction occurs. The contradictions can be either very different sentiments expressed on the same topic or sentiment shifts (i.e., change of polarity of the sentiments expressed on some topic); they can be identified by examining the index structure at different time granularities 0. The final result is a stream of such contradictions, which are represented as peaks in the data stream shown in the bottom right of Fig. 5.

6. DATA STREAM USE CASE

Problem P7 is concerned with streams of data representing measures, which need to be first chunked into working sets using windows, then summarized, and then analyzed for identifying outliers.

6.1 Common Pattern Types

The datasets to be analyzed consist of streams of several measures produced by different sensors. The corresponding pattern types are shown below.

Measure: [sid, sensor,{oid, variable, value}]
MeasureStream:[stid,<Measure, timestamp>]

6.2 Model of P7 Case

Problem P7 is solved by a pipeline of Mega-Modules: the first one is the windowing Mega-Module, which chunks the stream using a sliding window, the second one computes a summary of the values in the window, and the third one performs outlier detection (see Fig. 6).

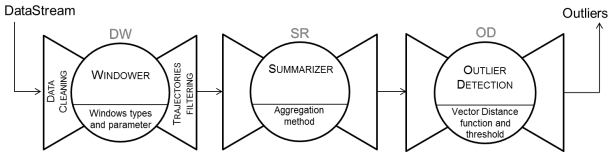


Figure 6: Problem P7

- **SUMMARIZER** (SR) – builds a concise summary of the multidimensional features associated to a set of given objects (e.g., based on histograms, or kernel density estimators).
- **OUTLIER DETECTOR** (OD) – given a population of objects described by multidimensional features and a notion of distance in the feature space, this module identifies an object as an outlier when the vector of the object differs significantly from the median vector of the population, or when the local neighborhood of the object is significantly less dense than its extended neighborhood.

After modeling P7, we realized that the outlier detection module performs a task that is very similar to the post-processing of the Trajectory Reconstruction (TR) module of P1. Indeed, once a trajectory is defined as an appropriate vector of features, a trajectory is outlier when its features significantly diverge from the features of the median. Then a remodeling of P1 takes place, by eliminating the post-processing from TR and chaining OD between TR and TM. This experience gives an indication of how we expect mega-modeling to evolve, with both top-down problem decompositions and bottom-up identification of reusable components across problems.

7. MEGA-MODULES IN THE CLOUD

In the following we show how to transform Mega-Modules into executable programs that use the Map/Reduce (M/R) paradigm, and could be executed on systems such as Hadoop, Dryad, or Stratosphere [10]. In particular, we use the Stratosphere platform, which provides two different programming models:

1. The PACT programming model, which supports flow programs based on second order functions (such as Map and Reduce) and User Defined Functions (UDFs). For example, the relational SELECTION operator is modeled by a MAP together with an UDF removing the input tuples that do not satisfy a filter predicate.
2. For higher level programming, Stratosphere provides a programming framework called SOPREMO which allows programmers to define custom packages, the respective operators and their instantiation.
3. The METEOR language allows programmer to write programs by specifying sequences of Jason-like statements using built-in operators or operators from one or more SOPREMO packages. Each METEOR statement may refer to output variables of previous statements or to input variables.

Once a METEOR program has been fully specified including data sources and data sinks, a compiler compiles it into a PACT program by:

- replacing library specific operators by their corresponding instantiations (part of the library);
- "chaining together" all partial PACT programs according to the variables of the Meteor program.

The result of this compilation is an executable PACT program which then may be optimized and compiled further for execution in a cluster based environment. Similar transformations are possible for target execution systems such as Hadoop (using the languages PIG or Hive) or Dryad (using the SCOPE language).

Fig. 7 shows a fragment of METEOR program which implements the Mega-Module TR (TRAJECTORY RECONSTRUCTION), which uses functions of previously defined SOPREMO packages. The program reads observations from the input, then cleans the observations that are considered as outlier or inconsistent, then builds histories as sequences of observations, then reconstructs trajectories using the StartStopFunction that is provided as local input to the TrajectoryReconstruction function, and finally writes trajectories to the output; note that outlier filtering is omitted from TR and associated with the Mega-Module OD. Mega-Modules OUTLIER DETECTION (OD), TRAJECTORY CLUSTERING (TM) and MOBILITY PROFILE (MP) can be encoded by similar METEOR programs.

```
MegaModule TR {  // Trajectory Reconstruction

using Trajectory ; //library used

input $ObservationData ;
input function $TStartStopDef  from Trajectory.StartStopFuntion;
output $TrajectoryData ;


$Observations = read from input $ObservationData;
$CleansedObservations = Cleanse $Observations
$Histories = BuildHistory $CleansedObservations
$Trajectories = TrajectoryReconstruction $Histories
                using function $TStartStopDef;
write $Trajectories to output $TrajctoryData ;

} //end MegaModule TR
```

Figure 7: METEOR Program Fragment for TR

Problem P1 was used in an application context where similar profiles of citizen are matched so as to propose car-pooling options to them. Such an extension can be supported as follows. First, observation data are extended with the identity of citizens. Then the high-level computation is a METEOR program that, for each user, invokes the function ComputeProfile that embodies the Mega-Modules TR, OD, TC and MP, and then calls a match function that produces pairs of candidate commuters. A simplified version of the corresponding METEOR program fragment is:

```
$CitizenData = read from input $ObservationData
$CitizenCommutes =
   Group CitizenData by CitizenID into
      {(CitizenID, Profile:
            ComputeProfile (CitizenData.Observations)};
$Candidates = Match $CitizenCommutes;
write $Candidates to output CandidateCommuters.
```

Finally, note that current M/R frameworks, including Stratosphere, do not support streams; however, windows allow gathering large collections of information during suitably long periods of time; thus, each window can be processes as a separate batch. In this way, it is possible to manage stream problems, such as P4-P7.

ACKNOWLEDGMENT. This work is supported by the Search Computing (SeCo) Project, sponsored by ERC, and by the FET-Open project DATASIM.

8. REFERENCES

[1] P. Ball: Why Society is a Complex Matter, Springer-Verlag,2012.

[2] S. Ceri, E. Della Valle, D. Pedreschi, R. Trasarti: Mega-Modeling for Big Data Analytics, *ER-2012*, LNCS 7352, Springer, pp. 1-15, October 2012.

[3] R. Trasarti, F. Giannotti, M. Nanni, F. Pinelli: Mining mobility user profiles for car pooling. In ACM International Conference on Knowledge Discovery and Data Mining, KDD 2011.

[4] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli: Trajectory Pattern Mining. In ACM International Conference on Knowledge Discovery and Data Mining, KDD 2007.

[5] S. Rinzivillo, S. Mainardi, F. Pezzoni, M. Coscia, F. Giannotti, D. Pedreschi: Discovering the Geographical Borders of Human Mobility. KI - Künstliche Intelligenz, 2012.

[6] M. Balduini and E. Della Valle: Tracking Movements and Attention of Crowds in Real Time Analysing Social Streams: The case of the Open Ceremony of London 2012. Semantic Web Challenge, 2012.

[7] M. Balduini, I. Celino, D. Dell'Aglio, E. Della Valle, Yi Huang, T. K. Lee, S. H. Kim, V. Tresp: BOTTARI: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. J. Web Sem. 16: 33-41, 2012.

[8] M. Tsytsarau, T. Palpanas, K. Denecke: Scalable Discovery of Contradictions on the Web. International World Wide Web Conference (WWW), Raleigh, NC, USA, April 2010.

[9] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos: Online Outlier Detection in Sensor Data Using Non-Parametric Models. Proc. VLDB, Seoul, Korea, September 2006.

[10] A. Heise, A. Rheinlander, M. Leich, U. Leser, F. Naumann, Meteor/Sopremo: An Extensible Query Language and Operator Model, Workshop on End-to-End Management of Big Data, Istanbul, Turkey (2012).

[11] Tomasz Imielinski, Heikki Mannila: A Database Perspective on Knowledge Discovery. Communication of the ACM 39 (11): 58-64, 1996.

[12] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo and R. Trasarti: Unveiling the Complexity of Human

Mobility by Querying and Mining Massive Trajectory Data. *The VLDB Journal* 20(5): 695-719, 2011.
[13] G. Wiederhold, P. Wegner, S. Ceri: Towards Mega-Programming, ACM Communications, 35:11, 1992.
[14] J. Bezivin, F. Journault, P. Valduriez: On the need for Megamodels, OOPSLA2004/GPCE Workshop.
[15] J-M. Favre, T. Nguyen: Towards a Megamodel to Model Software Evolution Through Transformations. Electr. Notes Theor. Comput. Sci. 127(3), 59-74, 2005.