Mega-modeling for Big Data Analytics

Stefano Ceri[1], Emanuele Della Valle[1], Dino Pedreschi[2], and Roberto Trasarti[3]

[1] DEI, Politecnico di Milano, via Ponzio 34/5, 20133, Milano
[2] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo, 3, 56127 Pisa
[3] ISTI-CNR, Istituto di Scienze e Tecnologie dell'Informazione del CNR, Pisa

Abstract. The availability of huge amounts of data ("big data") is changing our attitude towards science, which is moving from specialized to massive experiments and from very focused to very broad research questions. Models of all kinds, from analytic to numeric, from exact to stochastic, from simulative to predictive, from behavioral to ontological, from patterns to laws, enable massive data analysis and mining, often in real time. Scientific discovery in most cases stems from complex pipelines of data analysis and data mining methods on top of "big" experimental data, confronted and contrasted with state-of-art knowledge. In this setting, we propose **mega-modelling** 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.

1 Introduction

The grand challenge of modern scientific data processing is the ability to use "big data" for knowledge discovery. Progress in key areas - such as social and economic resilience, health, transportation, energy management - depends on a strategic use of data, e.g., for understanding disease spreading or economic crises, for energy distribution policies which make the best use of resources, for on-line alerting systems that take into account traffic, road conditions, hazards, and so on. Decision makers are thrilled by the possibility of anticipating the impacts of different possible decisions, i.e., exploring various future scenarios at different degrees of detail, employing a variety of predictive methods (such as multi-level and micro-macro models, patterns, simulations and what-if analyses).

As advocated by the visions of FuturICT [1], the FourthParadigm [2] and Haas et al. [3], this challenge cannot be addressed by simply deploying currently available technology; it entails a profound innovation of ICT research, by boosting a reformulation of all core information technologies in terms of a global techno-social ecosystem, where ICT opens to the challenges of supporting the complexity of scientific computational models. We need modelling capabilities that leverage on the power of big data, e.g., conceiving what-if models and scenarios that realistically portray the outcomes of possible interventions or changes onto complex socio-economic phenomena. Mastering such new scenarios requires establishing open platforms where

P. Atzeni, D. Cheung, and R. Sudha (Eds.): ER 2012, LNCS 7532, pp. 1–15, 2012.

scientists and developers can integrate and compose deep models, big-data-driven analytics and agent-based simulations and create empirically validated, computationally replicable modelling components.

But, beyond and before technological platforms, we need *a comprehensive theory blending simulation models, analytical models, ontological models and data-driven models into one picture*. Modelling, as we know it today, is required to scale up to a higher level, that we call **mega-modelling**: a comprehensive theory and technology of model construction (with an emphasis on incremental approaches), model search, model fitness evaluation, model composition, model reuse and model evolution. We need entirely new *model 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.

This paper aims at making a first step in the above direction, by introducing the concept of **mega-modelling** and then some abstractions for mega-model composition, attempting both a top-down definition and a bottom-up recognition of these abstractions in previous projects of the authors. The remainder of the paper is organized as follows. Section 2 traces the origin of this research, by positioning it relative to model-driven research in software engineering and to inductive data mining. Section 3 provides a definition of mega-modules and a preliminary view on its composition abstractions. Section 4 introduces mega-schemas and patterns as fundamental ingredients of mega-modules. Finally, Section 5 provides some examples which trace, in existing work, the presence of mega-modules ancestors.

2 Scientific Pillars of Mega-Modelling

The roots of mega-modelling can be traced to an article appeared in 1992 on "Mega-programming" [4]; mega-programs represent large, autonomous computing systems whose interfaces are described through a data-centric approach and whose execution behaviour can be inspected. After about one decade, Bezivin et. al. in 2004 introduced the term mega-model to denote modelling large software components [5], thus paving the road to the school on Model-Driven Engineering (MDE) that developed thereafter [6]. Then, in this community, mega-modelling has been mostly used as a term for denoting the higher-order relationships between models (such as *representationOf, conformsTo, isTransformedIn)* [7], or, more recently, for tracing the dependencies between models during their evolution [8]. However, the innovative aspects of mega-modelling go beyond classical model-driven software generation, as 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. This is the new and enhanced meaning that we give to mega-modules, by reconsidering them from the perspective of "big" data analytics.

In the database/data mining community, the idea of a unifying approach towards data analysis and mining has been around for several years, since the seminal paper on *inductive databases* and *data mining as a querying process* by Imielinski and Mannila [9]; a fundamental aspect of the approach 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; it is used in [10] and [11], where extracted data patterns are defined as views on top of data tables, and as such can be composed with domain-specific data representing genetic information [10] or spatio-temporal trajectories expressing human mobility [11]. Shaping computational results through regular table formats has recently found another intriguing field of application in the context of social computations, where tabular formats have been elected to drive interactions with human crowds through crowd-sourcing [12] and crowd-searching [13]; the above models hint to a possible seamless extension of scientific computation to crowd-based computations.

3 Mega-modules for Scientific Big Data Processing

In the context of this paper, a **mega-module** is 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).

Let us propose a running example inspired to mobility data mining in M-Atlas [*11*]*, where a mega-module uses positions of mobile users to infer how big masses of people move from regions to regions – thereby inferring how persons move at an aggregate level. Positions as detected by GPS systems in mobiles and reported in the territory with an associated timestamp represent the input data. Information about streets and roads can be considered as invariant. Moving points are recognized as trajectories, a known pattern featuring a starting point, an ending point, and a sequence of intermediate points. Trajectories are clustered and aggregated by the mega-module, whose output reports, in a compact way, the most significant "flock movements" of groups of people.*

3.1 Phases of Mega-module Computations

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.
  In the running example, several observations of the positions of the same moving object are assembled into a single trajectory.
- 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.
  In the running example, trajectories are assembled and reported as movements of groups of people (flocks).
- 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 the running example, reported flocks have a population above a given threshold and connect specific portions of territory, e.g. recognized as regions in a map.

In **Fig. 1**, we propose (on the left) a mega-module graphical element, which visually captures the characteristics of a mega-module as described above, and (on the right) we show how to represent the running example with such a graphical element. The partitioning of mega-module activity in three phases is used in defining its compositional properties, as described next.

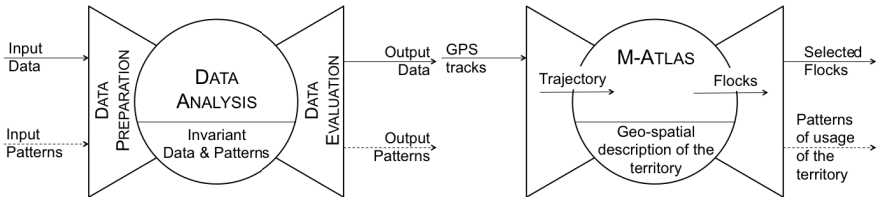


Fig. 1. Visual presentation of a generic mega-module and of the running example

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.

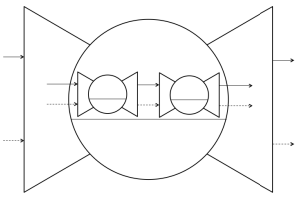
A mega-module inspection consists in extracting its controls asynchronously, during its execution; this in turn allows the enclosing environment to trace mega-module execution, to estimate completion time, and to anticipate the quality of its results. We regard the data and patterns that may be exchanged 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.

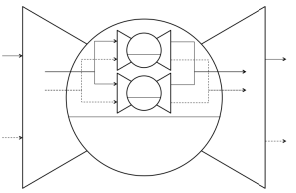
3.2 Composition Abstractions

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. Below, we present an initial set of abstractions; they are orthogonal, but most likely incomplete, and further investigation is needed to consolidate them.

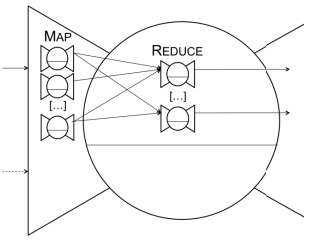
General-Purpose Composition Abstractions. The first three abstractions apply to arbitrary mega-modules and characterize classical ways of partitioning computations, through pipelines and parallel control and by fragmenting computations.

Pipeline decomposition. The simplest form of control is a pipeline that occurs when the output of one mega-module feed the input of another mega-module. A pipeline can be abstracted by singling out the prepare phase of the first one and the evaluation phase of the last one, and, then, compounding the internal chain of analysis-evaluation-preparation-analysis as a two-step analysis phase; this generalizes to n-step pipelines.



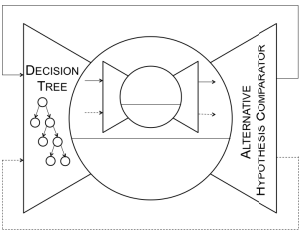


Parallel decomposition. The next form of control is a parallel flow, which occurs when the input data after preparation can feed two or more distinct analysis activities, each of which can be embodied by a mega-module. Each analysis activity may itself require local preparation and evaluation; therefore the parallel coordination abstraction entails the parallel execution of several different mega-modules, with their own preparation and evaluation, in the context of an enclosing mega-module with one prepare and one evaluation phase. During the evaluation phase of the enclosing mega-modules, results objects from each internal mega-module are composed to generate global result objects, according to a variety of possible mechanisms, e.g. they can be sorted into a list of result objects, can be put in one-to-one correspondence and then returned identically or composed.



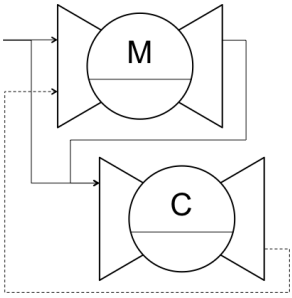
Map-reduce decomposition. The most important aspect of computation over big data is the support of parallelization, that consists of solving a complex problem over large data sets by parallel execution of less complex problems, typically over portions of the initial data set. The most popular parallelization paradigm, map-reduce [14], consists of a mapping abstractions that assigns portions of the computation to distinct processing units, each generating a local result, and a reduction abstraction that is capable of computing a global result from the local results. In terms of mega-modeling, the map part should occur during data preparation and should manipulate input data and patterns into a format that exposes input objects suitable for parallelization; the reduce part should occur during data analysis and produce global output objects from independently computed result objects, while the evaluation is responsible of result post-processing and presentation to users. The ability of algorithm designers consists of expressing computations by understanding how they can be twisted to expose both their map and reduce components.

Specific Composition Abstractions. The following are special cases of composition abstractions that we have recognized as typical analytical processes in existing systems.

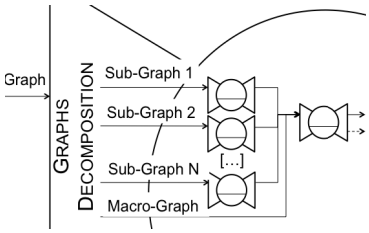


What-if control. A classical way of mining big data is to explore many alternative solutions that would occur for different choices of initial setting of models and/or parameters. Essentially, this control abstraction is a form of iteration driven by an analytical goal, allowing to *repeat* a mega-module 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.



Drift control. 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.



Component-based graph decomposition. Many mega-module computations apply to input data representing (large) networks and graphs and this makes parallelization 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 of each sub-network to an internal mega-module and then integrating the results using one additional combination mega-module. Interestingly, the emerging field of network science has demonstrated that many complex networks, and notably social networks, exhibit a non-random structure with a tendency of nodes to cluster into tightly knit communities (many edges among nodes in the same community), while different communities are bridged by relatively few long-range edges: a structure that account for the small-world phenomenon (any two persons are only a few hops away from each other in the social network) and for the tendency of social triangles to form (two persons sharing common friends have a higher chance of becoming friends of each other) [18,19,20]. As a consequence, many real networks from different domains (social, web-related, biological, technological) can be partitioned into components or communities, and a wide variety of methods for community detection has flourished [21]. Treating each sub-network independently can solve a modular sub-problem in a parallelizable way; arcs connecting the sub-networks are then either disregarded or considered by a separate mega-module, which produces computed objects to be assembled and evaluated at the end. Community structure is therefore also a way of separating a micro-scale analytical process (within a community) and a meso- or macro-scale analysis (among different communities, taking each community as a node in a higher level network).

4 Data Management through Schemata and Patterns

Data-intensiveness is the characteristic feature of big data analytics, therefore it is not surprising that data and patterns are given an essential role.

4.1 Mega-schema

We define a *Scientific Data Experiment* as a unitary data research experience over a scientific big data collection, where the experiment can possibly span through years, over different laboratories, involving a huge number of data sources, each with massive amount of data. Then, we make the assumption that all the data used in the input and output schemas of the experiment must be conformant to a unique **mega-schema**.

Methodologically, the global schema design should be the first task in setting a scientific data experiment. This step should be heavily influenced by the agreed ontologies of the domain under study: schema names should reflect the agreed names as codified in the ontology. Ontology-driven schema design and annotation methods are already in use in many scientific communities, e.g. medicine and biology, and we advocate that a similar trend should take place in all cases of big data analytics.

We do not make assumption on the specific mega-schema syntax – candidates are relational, XML, and RDF – and semantics – candidates are ER, XML Schema, and OWL. Establishing a mega-schema in a context of conflicting/pre-existing data and computing resources requires the availability of a variety of data conversion tools, due to the intrinsic heterogeneous nature of those resources; we assume the a-priori establishing of one mega-schema, forcing each data source to be conformant using a "global as view" (GAV) mapping, with the strong belief that such practice will result, in the long run, beneficial also in terms of the overall data conversion complexity.

4.2 Patterns Organization

Patterns describe data regularities, which are sometimes exposed at data definition and in most times inferred as result of data analysis and mining; their format is domain-independent, therefore the "schema" of patterns reflects the underlying structure of the discovery that the scientific experiment should produce, rather than its input and output data. We are henceforth assuming that there exists a finite number of pattern structures capable of describing all the forms of regularity that are worth extracting through a given scientific data experiment. We assume patterns to describe large numbers of Items, all with the same format; Items are structured objects with a schema, and can be typed (ItemType is a label of the type). Patterns can be described by means of type constructors with Items and numerical attributes expressing their properties (either exact or approximate). **Fig. 2** lists examples of patterns from a broad spectrum of problems, although at a very shallow level of details, as each pattern should be further specialized by considering the specific data analysis experiment.

CLASSIFICATION. The computation extracts classes from a population based on some classification algorithm operating upon its property, and then computes statistics – from simple frequencies up.

Data: Population(Item)
Pattern: Class(Name, AggrStats)

CLUSTERING. The computation operates upon a collection and extracts its clusters, where each cluster has a name, an extent consisting of its elements, possibly a centroid element, and then statistics – from cardinalities up.

Data: Collection(Item)
Pattern: Cluster(Name, Extent: [Item], CentroidItem, AggrStats)

STREAMING. The computation aggregates data from a stream, by aggregating those items of a given type and within a given time interval, typically the most recent, and then computing aggregate properties.

Data: Stream(TimeStamp, Item)
Pattern: StreamStats(ItemType, TimeInterval, AggrStats)

STREAMING WITH WINDOWS. The computation aggregates data from a stream which is subdivided in windows, by aggregating within each window those items of a given type and then computing aggregate properties.

Data: Stream(Window, StartTimeStamp, EndTimeStamp, Content:[Item])
Pattern: WindowedStats(Window, ItemType, AggrStats)

TREE. Classical computations provide the descendants or ancestors of a given node, or classify a new node relative to an existing hierarchical taxonomy, e.g. by showing the path from the root to the node which is most similar to the new node.

Data: Tree (Item, Descendent, ChildItem)
Pattern: Descendants(Item, To: [Item])
Ancestors(Item, From: [Item])
Classify (Item, Path[Item])

GRAPH. Classical computations provide a decomposition of a graph into components by minimizing the edges which interconnect nodes of different components, or find the "friend" nodes which are at a given "nearness" from a given node.

Data: Graph(FromItem, ToItem)
Pattern: Components(Name, Components: [Node])
Friends(FromItem, NearnessLevel, To: [Item])

DISTANCE-GRAPH. If the graph includes a label expressing node distances, a classic computation find the shortest path between any two items, exptessed as a sequence of nodes connecting them and a totaldistance.

Data: D-Graph(FromItem, ToItem, Distance)
Pattern: ShortPath(OriginItem, DestinationItem, Path: [Item], TotalDistance)

ASSOCIATION RULES. A classical data mining problem is to find items which appear together within a basket; an association rule has an head and a body describing item sets, and then statistical properties of support and confidence defining the rule's interest.

Data Basket(Item)
Pattern: Rule(Head:[Item], Body:[Item], Support, Confidence)

MOVING POINTS. When a system accumulates indications of positions from individuals, a classical computation is the reconstruction of the trajectories, i.e. the sequence of locations which are traversed by the same item from an initial location to a final location.

Data: Point(Item, Time, Location)
Pattern: Trajectory(Item, FromLocation, ToLocation, Steps:[Location], StepCount: Number)

FLOCKS. When a system accumulates multiple trajectories, another classical computation is the combining of trajectories together to recognize flocks, i.e. movements of groups of individuals across regions.

Data: Trajectory(Item, FromLocation, ToLocation, Steps:[Location], StepCount: Number)
Pattern: Flock(FlockName, FromRegion, ToRegion, Objects: [Items], ObjectCount: Number)

Fig. 2. Examples of patterns from a broad spectrum of problems

5 Examples

5.1 Parallel and Pipe

As an example of usage of parallel and pipe composition we illustrate BOTTARI [15] – an augmented reality application for personalized and localized recommendations of points of interest, experimentally deployed for recommending restaurants in the Insadong district of Seoul. At a first look, it may appear like other mobile apps that recommend restaurants, but BOTTARI uses inductive and deductive stream reasoning [16] to continuously analyze social media streams (specifically Twitter) to understand how the social media users collectively perceive the points of interest (POIs) in a given area, e.g., Insadong's restaurants.

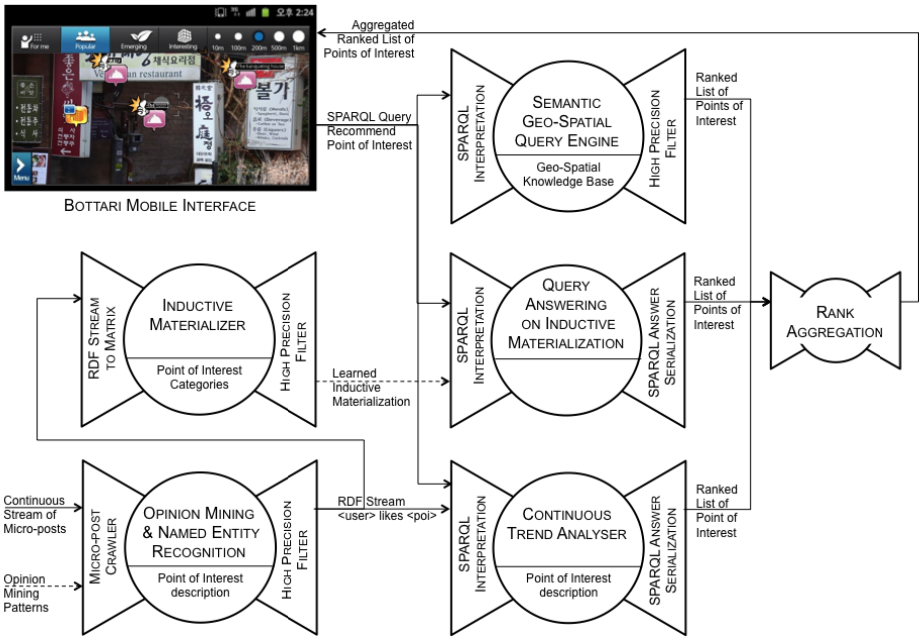


Fig. 3. The Mega-modeling of BOTTARI

BOTTARI was developed on LarKC platform [17], but the plug-able nature of LarKC and its orientation to scientific workflows, allows to cast BOTTARI in the Mega-modeling framework. In **Fig. 3**, we present a redesign of BOTTARI's interaction workflow that is twisted towards the mega-modules concept. The Micro Post Crawler continuously streams 3.4 million tweets/day related to Seoul in the Opinion Mining & Named Entity Recognition analytical sub-module that, knowing all Insadong restaurants, identifies the subset related to the Insadong restaurants (thousands per day) and detects the users' opinions. The result is an RDF stream of positive, negative and neutral ratings of the restaurants of Insadong. It flows at an average rate of a hundred tweets/day, peaking at tens of tweets/minute. The stream is

processed in real-time by the INDUCTIVE MATERIALIZER and by the CONTINUOUS TREND ANALYZER. The former transform the data in a matrix (users x restaurants) and incrementally maintain its inductive materialization using the internal knowledge about restaurant categories. The latter incrementally identifies the top restaurants in the week, month, quarter, etc. Whenever a user ask BOTTARI for a recommendation, a SPARQL query is sent to three Mega-modules: SEMANTIC GEO-SPATIAL QUERY ENGINE, which returns a list of restaurants that matches the semantic criterion requested by the user (e.g., traditional cuisine) ordered by distance from the user; the QUERY ANSWERING ON INDUCTIVE MATERIALIZATION, which uses the patterns identified by the INDUCTIVE MATERIALIZER to return a list of restaurants order by the probability that the user likes them; and the CONTINUOUS TREND ANALYZER, which return a list of restaurants ordered by number of positive ratings in a given time window (e.g., the last week). A RANK AGGREGATOR mega-module combines the lists and returns the recommendations to BOTTARI interface.

5.2 Mega-modelling of a Proactive Car-Pooling Service

A more complex analytical process is depicted in **Fig. 4**, which supports the intelligent, proactive car-pooling system of [22]. At the level of each individual mobile users, trajectories corresponding to actual trips are reconstructed, and then clustered to find the (systematic) routines of the user, e.g., her home-work-home commutes or frequent trips for bring-get activities, such as bringing kids to school. For each cluster a typical representative trip is extracted through a parallel decomposition, and a concept-drift mechanism is used to monitor when user's routine change, e.g., as a result of moving or the arrival of a new-born baby. The individual routines are, then, aggregated at collective level, according to two different purposes: to find the patterns of systematic mobility and study the city access paths (bottom right), and to find best matches among pairs of commuters with systematic routines, that can be put in contact for car-pooling. In our study over the city of Pisa, we found that there is a car-pooling potential of 67% (i.e., every routine has a 67% chance of being served from a matching routine of another user, see [22]), showing that car-pooling systems based on big data analytics have potentially a high impact in reducing systematic traffic.

5.3 Drift Control

The mega-model in Fig. 5 depicts a system that leverage the analysis of tourist routes in an area of interest for the twofold purpose of understanding aggregated touristic behavior and recommending popular routes to individual tourists. Trajectories recording tourist visits are reconstructed at the individual level and then aggregated; trajectory pattern mining is then used to discover the frequently followed routes. The discovered patterns are delivered both to the tourism authorities for analytical purposes and to the individual tourists in form of recommended popular routes. A concept-drift composition is used to monitor the validity of the discovered patterns against the incoming stream of tourist trajectories, as well as the emergence of novel patterns.

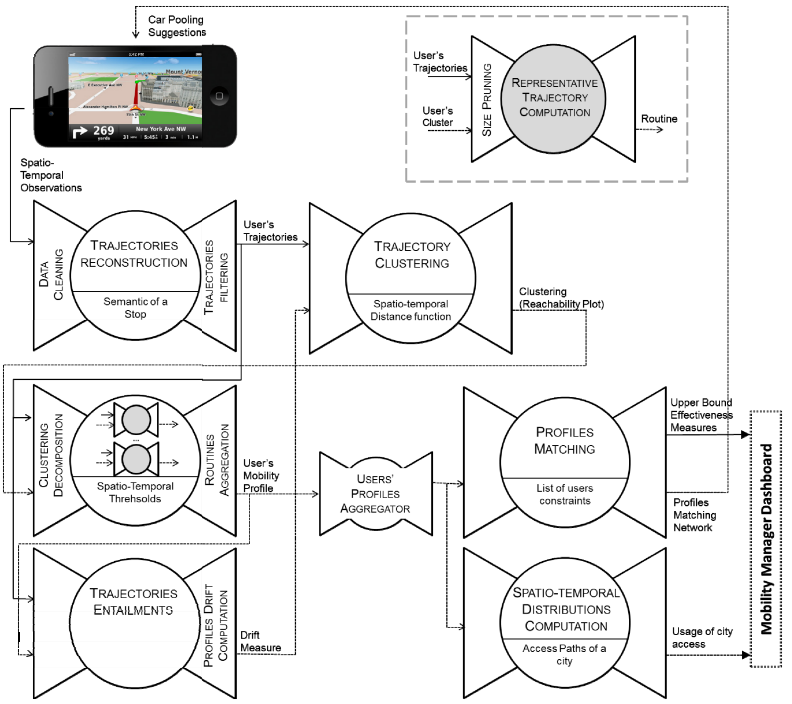


Fig. 4. The Mega-modeling of an intelligent car-pooling service

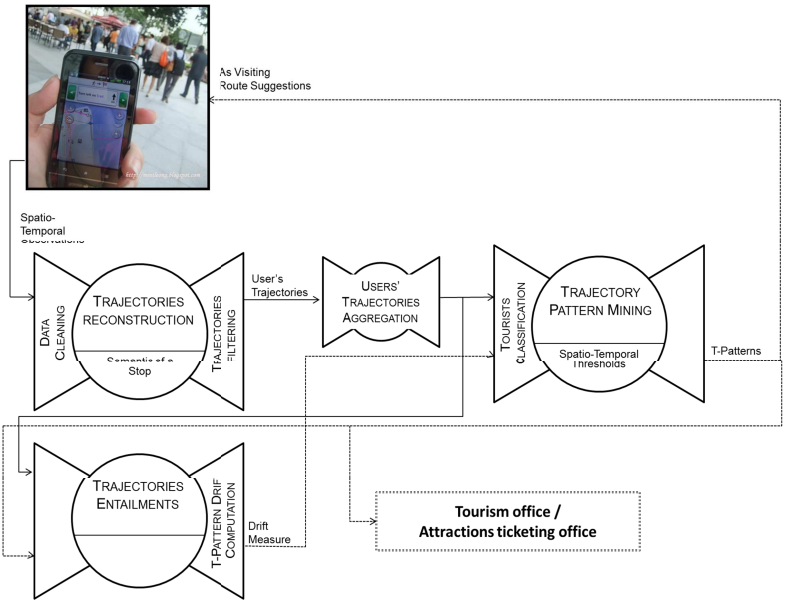


Fig. 5. The Mega-modeling of a touristic recommendation service

5.4 Component-Based Graph Decomposition

An example of component-based graph decomposition is depicted in Fig. 6, based on the idea of discovering the geographical borders delimiting the real mobility basins dictated by the big data of human mobility [23].

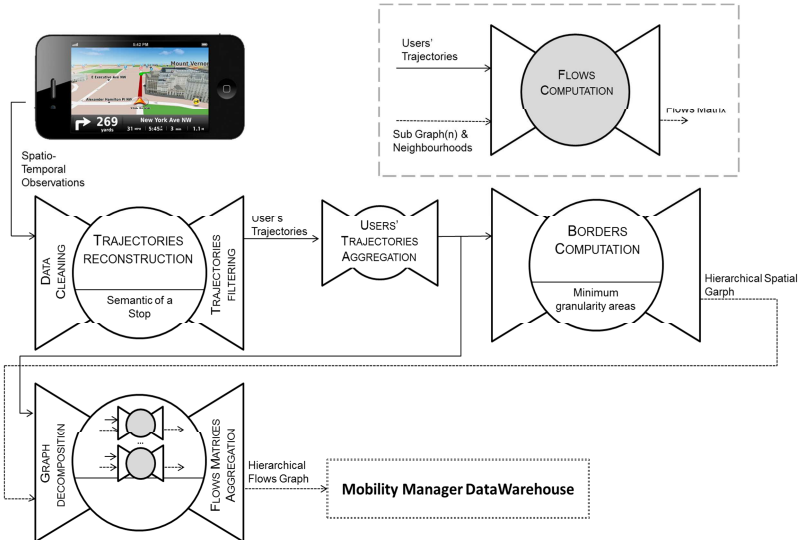


Fig. 6. The Mega-modeling of a micro-macro decomposition of mobility analysis

The key mega-module here is the *border* computation, illustrated in Fig. 7. Starting from a given zoning of the territory, tessellated into census zones, the preparation phase constructs a network whose nodes are the zones and the weighted edges between any two zones represent the number of travels originating in the first and ending in the second. The analysis phase consists in discovering densely connected sub-graphs in this network by means of a community detection method, thus highlighting groups of zones that are highly connected by many travels compared to the lower connectivity among different modules. The evaluation phase consists in mapping these modules back to geography, and drawing appropriate borders to delimit the different modules. The borders mega-module is a means to highlight a hierarchical structure in a large network, thus separating a micro level (within each separate module) and a macro-level (where each module is abstracted to a node and the links among different modules are considered). The mega-modeling of Fig. 6 then exploit this decomposition for a parallel (or map-reduce) computation over the separate modules of mobility analysis, such as origin-destination flow matrices or more sophisticated pattern mining, as in previous examples.

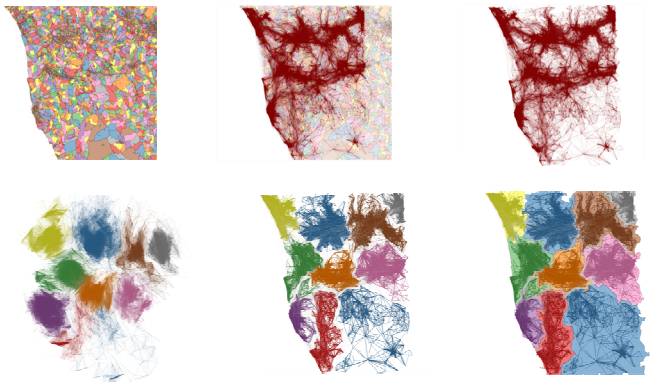


Fig. 7. The steps of the *borders* computation (from top left to bottom right): tessellation into census zones; construction of the flow network between any two zones; forgetting geography; discovery of modules (communities) in the network; mapping of communities back to geography; delimitation of modules so as to discover the borders dictated by human mobility [23]

6 Conclusions and Future Work

This paper introduces the concept of mega-modelling and some abstractions for mega-model composition; its objective is to raise the interest of the community of scientific big data processing on model composition and reuse. Our approach is very preliminary, and needs formalization and extensions, e.g. each model should be associated with a meta-model explaining the model's syntax and behaviour, and therefore mega-modelling should be associated with suitable languages for expressing meta-model aspects. In a broad vision, meta-models should be indexed and searchable, and model interoperability should be supported, so as to build a world-of-modelling platform, capable of supporting the creation of complex analytical and simulation processes by composing knowledge discovery mega-modules. Different language and visual interfaces to mega-modelling should also be devised, so as to empower different classes of users with varying levels of expertise (data scientists, social scientists, policy makers, ordinary citizens) with appropriate means for creating realistic and understandable simulations and what-if analyses. In our future work, we plan to further develop the mega-modelling abstractions in the context of large inter-disciplinary projects such as FuturICT and Genomic Computing.

References

1. Bishop, S., Helbing, D.: FuturICT Project Summary, `http://www.futurict.eu`
2. Hey, T., Tansley, S., Tolle, K. (eds.): The Fourth Paradigm. Data-Intensive Scientific Discovery. Microsoft Research (2009)
3. Haas, P.J., Maglio, P.P., Selinger, P.G., Tan, W.-C.: Data is Dead ...Without What-If Models. In: Proceedings of the Very Large Data Bases Endowment, PVLDB, vol. 4(12) (2011)

4. Wiederhold, G., Wegner, P., Ceri, S.: Towards Mega-Programming. ACM Communications 35, 11 (1992)
5. Bezivin, J., Journault, F., Valduriez, P.: On the need for Megamodels. In: OOPSLA 2004/GPCE Workshop
6. Favre, J.-M., Nguyen, T.: Towards a Megamodel to Model Software Evolution Through Transformations. Electr. Notes Theor. Comput. Sci. 127(3), 59–74 (2005)
7. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer 39(2), 25–31 (2006)
8. Seibel, A., Neumann, S., Giese, H.: Dynamic Hierarchical Megamodels: Comprehensive Traceability and its Efficient Maintenance. Software and System Modeling 9(4), 493–528 (2010)
9. Imielinski, T., Mannila, H.: A Database Perspective on Knowledge Discovery. Communication of the ACM 39(11), 58–64 (1996)
10. Blockeel, H., Goethals, B., Calders, T., Prado, A., Fromont, E., Robardet, C.: An Inductive Database System Based on Virtual Mining Views. Data Mining & Knowledge Discovery 24(1), 247–287 (2012)
11. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., Trasarti, R.: Unveiling the Complexity of Human Mobility by Querying and Mining Massive Trajectory Data. The VLDB Journal 20(5), 695–719 (2011)
12. Franklin, M.J., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: CrowdDB: Answering Que-ries with Crowdsourcing. In: Proc. ACM-Sigmod, Athens (June 2011)
13. Bozzon, A., Brambilla, M., Ceri, S.: Answering Search Queries with Crowdsearcher. In: Proc. WWW 2012, Lyon (April 2012)
14. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Operating Systems Design and Implementation (USDI 2004), pp. 137–147 (2004)
15. Celino, I., Dell'Aglio, D., Della Valle, E., Huang, Y., Lee, T., Park, S., Tresp, V.: Bottari: an Augmented Reality Mobile Application to deliver Personalized and Location-based Recommendations by Continuous Analysis of Social Media Streams. J. Web Semantics (to appear, 2012)
16. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Huang, Y., Tresp, V., Rettinger, A., Wermser, H.: Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics. IEEE Intelligent Systems 25(6), 32–41 (2010)
17. Assel, M., Cheptsov, A., Gallizo, G., Celino, I., Dell'Aglio, D., Bradesko, L., Witbrock, M., Della Valle, E.: Large Knowledge Collider: a Service-oriented Platform for Large-scale Se-mantic Reasoning. In: Proc. WIMS 2011 (2011)
18. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393, 440 (1998)
19. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509 (1999)
20. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press (2010)
21. Coscia, M., Giannotti, F., Pedreschi, D.: A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining 4(5), 512–546 (2011)
22. Trasarti, R., Pinelli, F., Nanni, M., Giannotti, F.: Mining mobility user profiles for car pooling. In: KDD 2011, pp. 1190–1198 (2011)
23. Rinzivillo, S., Mainardi, S., Pezzoni, F., Coscia, M., Pedreschi, D., Giannotti, F.: Discovering the Geographical Borders of Human Mobility. KI - Künstliche Intelligenz (2012)