Expert Systems With Applications 115 (2019) 543–556

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

BIGOWL: Knowledge centered Big Data analytics[☆]

Cristóbal Barba-González, José García-Nieto[*], María del Mar Roldán-García, Ismael Navas-Delgado, Antonio J. Nebro, José F. Aldana-Montes

Departmento de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Málaga 29071, Spain

ARTICLE INFO

Article history:
Received 5 April 2018
Revised 26 July 2018
Accepted 14 August 2018
Available online 23 August 2018

Keywords:
Ontology
Big Data analytics
Semantics
Knowledge extraction

ABSTRACT

Knowledge extraction and incorporation is currently considered to be beneficial for efficient Big Data analytics. Knowledge can take part in workflow design, constraint definition, parameter selection and configuration, human interactive and decision-making strategies. This paper proposes BIGOWL, an ontology to support knowledge management in Big Data analytics. BIGOWL is designed to cover a wide vocabulary of terms concerning Big Data analytics workflows, including their components and how they are connected, from data sources to the analytics visualization. It also takes into consideration aspects such as parameters, restrictions and formats. This ontology defines not only the taxonomic relationships between the different concepts, but also instances representing specific individuals to guide the users in the design of Big Data analytics workflows. For testing purposes, two case studies are developed, which consists in: first, real-world streaming processing with Spark of traffic Open Data, for route optimization in urban environment of New York city; and second, data mining classification of an academic dataset on local/cloud platforms. The analytics workflows resulting from the BIGOWL semantic model are validated and successfully evaluated.

1. Introduction

In accordance with the recent Gartner's report,[1] an emerging challenge in Big Data is to construct data-driven intelligent applications that capture and inject domain knowledge in the analytical processes, including context and using a standardized format. Context refers to all the relevant (meta)-information to support the analysis and to help interpreting its results. This will facilitate the integration (in a standardized way) with third parties' data, algorithms, business intelligence (BI) and visualization services.

The use of semantics as contextual information will enhance the analytical power of the algorithms, as well as the reuse of single components in data analytics workflows (Ristoski & Paulheim, 2016). Therefore, the development of ways to make the domain knowledge explicit and usable is needed to improve the data processing and analysis tasks. The Semantic Web technologies can be used to annotate not only the knowledge domain of the data, but also the analytics' meta-data (Keet, Ławrynowicz, d'Amato, Kalousis, Nguyen, Palma, Stevens, & Hilario, 2015), including: algorithms' parameters, input variables, tuning experiences, expected behaviors and taxonomies. This will facilitate the reuse and composition of Big Data analytics in a proper manner, as well as to enhance the quality of consumed and produced data.

In this regard, ontologies describe concepts, relationships, classes, individuals, formal logic axioms and objects of a particular domain (Gruber, 1995). The objects refer to entities and events (concepts) in the real world, and their relations represent the semantic links between these entities. A series of studies have been appearing in the last few years, in which ontological approaches are suggested to enhance Big Data analytics (Konys, 2016; Kuiler, 2014). However, they are presented as conceptual frameworks, still in an early stage of development, and mostly oriented to the specific domain of health system applications.

This motivates us to propose an ontology-driven approach to support knowledge management in Big Data analytics workflows. The proposed ontology is called BIGOWL (BIG data analytics OWL[2]

[☆] This work has been partially funded by Grants TIN2014-58304, TIN2017-86049-R (Spanish Ministry of Education and Science) and P12-TIC-1519 (Plan Andaluz de Investigación, Desarrollo e Innovación). Cristóbal Barba-González is supported by Grant BES-2015-072209 (Spanish Ministry of Economy and Competitiveness). José García-Nieto is the recipient of a Post-Doctoral fellowship of "Captación de Talento para la Investigación" Plan Propio at Universidad de Málaga.

[*] Corresponding author.
E-mail addresses: cbarba@lcc.uma.es (C. Barba-González), jnieto@lcc.uma.es (J. García-Nieto), mmar@lcc.uma.es (M.d.M. Roldán-García), ismael@lcc.uma.es (I. Navas-Delgado), antonio@lcc.uma.es (A.J. Nebro), jfam@lcc.uma.es (J.F. Aldana-Montes).

[1] https://www.gartner.com/doc/3656517/adopt-datadriven-approach-consolidating-infrastructure.

[2] OWL refers to the Web Ontology Language described in Section 2.1.

https://doi.org/10.1016/j.eswa.2018.08.026
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

ontology), which acts as a formal schema for the representation and consolidation of knowledge in Big Data analytics. Knowledge incorporation is in turn beneficial for an efficient algorithmic performance, by taking part in operator's design, parameter selection, human interactive and decision-making strategies.

Our scientific hypothesis is as follows: "*The semantic annotation of Big Data sources, components and algorithms can acts as a link to capture and incorporate the domain knowledge to guide and enhance the analytical processes*". In addition, the semantic annotation can provide the background for reasoning methods based on axiomatic and rule logic recommendations.

To test this hypothesis, a semantic model has been generated, which comprises an RDF[3] (Resource Description Framework) repository that follows the BIGOWL scheme. This repository can be queried by high level algorithms using SPARQL. The goal is to properly feed artificial intelligence procedures capable of guiding the design of Big Data analytics workflows.

As a proof-of-concept, we show how BIGOWL can be used to guide the design of real-world and academic analytic workflows. A first case study consists in optimizing vehicular routes based on New York real-time Open Data about urban traffic (average speeds of vehicles, traffic densities, etc.).[4] The data source is managed by streaming processing tasks (Kafka and Spark), after which they are optimized (jMetalSP[5]) and visualized. The second case study is a classification workflow modeled by using the popular Weka[6] library for data mining, as well as the BigML in-cloud service.[7]

The main contributions of this study are:

- The proposed ontology, BIGOWL, has been designed and implemented for the representation and consolidation of knowledge in Big Data analytics. It considers a large and complemented set of concepts, attributes and relationships that have been taken from Big Data ecosystem.
- A semantic approach has been implemented to annotate (i.e. to "semantize") all the involved meta-data from multiple data sources, processing components and analytic algorithms. The meta-data are integrated following the BIGOWL structure and stored in a common RDF repository.
- The semantic model is evaluated in the context of two realistic use cases: real-time routing calculation in urban traffic and classical classification with decision trees. The proof-of-concept lead us to test our initial hypothesis.

The remaining of this paper is structured as follows. In Section 2, background concepts and literature overview are presented. Section 3 presents current practices in Big Data analytics. Section 4 describes the semantic model, comprising the ontology, RDF repository, mappings and workflow composition assistant. Section 5 presents the use case for testing and validation. In Section 6, a series of discussions are included. Conclusions and future work are drawn in Section 7.

2. Background and related work

To make this paper self-contained, this section describes background concepts in the Semantic Web field. A review of the state of the art is also provided to point out the main differences of the related works with the proposed approach.

Table 1
Basic OWL-DL semantic syntax used to formally define the proposed ontology.

Descriptions	Abstract syntax	DL syntax
Operators	*intersection*$(C_1, C_2, \cdots, C_n)$	$C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$
	union$(C_1, C_2, \cdots, C_n)$	$C_1 \sqcup C_2 \sqcup \cdots \sqcap C_n$
Restrictions	for at least 1 value V from C	$\exists V.C$
	for all values V from C	$\forall V.C$
	R is Symmetric	$R \equiv R^-$
Class Axioms	*A partial*$(C_1, C_2, \cdots, C_n)$	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$
	A complete$(C_1, C_2, \cdots, C_n)$	$A \equiv C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$

2.1. Background concepts

- Ontology. In accordance with Noy, McGuinness et al. (2001), an ontology provides a formal representation of the real world. It defines an explicit description of concepts in a domain of discourse (classes or concepts), properties of each concept describing various features and attributes of the concept (properties) and restrictions on properties. Ontologies are part of the W3C standard stack of the Semantic Web.[8] An ontology together with a set of individual instances of classes constitutes a knowledge base and offer services to facilitate interoperability across multiple heterogeneous systems and databases.
- RDF. Resource Description Framework (McBride, 2004) is a W3C recommendation that defines a language for describing resources on the web. RDF describes resources in terms of triples, consisting of a subject, predicate and object. RDF Schema (RDFS) (Staab & Studer, 2013) describes vocabularies used in RDF descriptions.
- OWL. The Ontology Web Language is used to define ontologies on the Web, which extends RDF and RDFS, but adding a vocabulary. From a formal description, OWL is equivalent to a very expressive description logic DL, where an ontology corresponds to a Tbox (Gruber et al., 1993). In this sense, OWL-DL is syntactic description that gives maximum expressiveness while retaining computational completeness and decidability (McGuinness, Van Harmelen et al., 2004). In this work, we use OWL-DL syntax summarized in Table 1 to formalize the proposed ontology.
- SPARQL is a query language for easy access to RDF stores. It is the query language recommended by W3C (Harris, Seaborne, & Prud'hommeaux, 2013) to work with RDF graphs (Prud, Seaborne et al., 2006), then supporting queries and web data sources identified by URIs.
- SWRL. The Semantic Web Rule Language provides the OWL-based ontologies with procedural knowledge, which compensates for some of the limitations of ontology inference, particularly in identifying semantic relationships between individuals (Horrocks, Patel-Schneider, Bechhofer, & Tsarkov, 2005). SWRL uses the typical logic expression "*Antecedent*⇒*Consequent*" to represent semantic rules. Both antecedent (rule body) and consequent (rule head) can be conjunctions of one or more atoms written as "$atom_1 \wedge atom_2 \wedge \cdots \wedge atom_n$". Each atom is attached to one or more parameters represented by a question mark and a variable (e.g., ?x). The most common uses of SWRL include transferring characteristics and inferring the existence of new individuals (Grosof & Poon, 2004).[9]

[3] RDF in W3C https://www.w3.org/RDF/.

[4] https://www.data.cityofnewyork.us/Transportation/Real-Time-Traffic-Speed-Data/xsat-x5sa.

[5] http://www.jmetal.sourceforge.net/.

[6] https://www.cs.waikato.ac.nz/ml/weka/.

[7] https://www.bigml.com/.

[8] https://www.w3.org/standards/semanticweb/.

[9] https://www.w3.org/Submission/SWRL/.

2.2. Related work

In the last decade, there have been appearing a series of studies in which ontological approaches are defined to express the knowledge domain in data mining and optimization algorithms. A representative set of these works are compiled in a recent survey (Dou, Wang, & Liu, 2015), in which they are organized by categories of algorithms and applications: association rule discovery (Marinica & Guillet, 2010), classification (Allahyari, Kochut, & Janik, 2014) and clustering (Jing, Ng, & Huang, 2010). In these applications, semantics is used with different objectives, such as: to reduce the search space by specifying restrictions, to filter results in the post-processing stage, and to annotate the results of data mining processes.

Following with this research line, some recent works include ontologies to guide the processes in machine learning tasks. For example, in Pinto, Scioscia, Loseto, and Ruta (2015) and Roldán-García, García-Nieto, and Aldana-Montes (2017), two different ontologies are used in the classification process to infer inconsistencies between concepts by means of semantic reasoning. In Phan, Dou, Wang, Kil, and Piniewski (2015), an ontology-driven deep learning model is proposed to predict human behavior.

In the field of optimization, an interesting approach has been recently proposed in Yaman, Hallawa, Coler, and Iacca (2017), where the ECO ontology is defined to formally represent knowledge in evolutionary computation algorithms. This ontology can be used for suggesting strategies for solving optimization problems. At the same time, an OWL ontology has been proposed in Li, Yevseyeva, Basto-Fernandes, Trautmann, Jing, and Emmerich (2017) to model and systematize the knowledge of preference-based multi-objective evolutionary algorithms. These ontologies are validated in use cases focused on algorithmic and parameter selection in academic problems.

From a different point of view, a parallel line of research focuses on defining ontologies for the semantic annotation of data analytic workflows. The main objective is to model the input and output of algorithms involved in data mining and knowledge base discovery (KDD) workflows to generate valid compositions. To this end, several OWL ontologies such as: KDDONTO (Diamantini, Potena, & Storti), DMWF (Kietz, Serban, Bernstein, & Fischer, 2010) and KD (Záková, Kremen, Zelezny, & Lavrac, 2011), were proposed. However, they did not describe the problem domain, or those basic concepts (algorithm, type of analysis, task, dataset, attribute, etc.) that can be combined to define entities or constraints. In fact, these ontologies were not designed with the objective of optimizing the performance of the data mining algorithms, since they do not offer detail enough to provide support to what is known as meta-learning. In Nguyen, Hilario, and Kalousis (2014), meta-learning is defined as the KDD procedure to improve performance in data mining processes, using information collected during the experimentation phase of these algorithms. In this regard, the use of semantics is considered not only for the algorithmic composition, but also for the improvement of data mining processes, taking advantage of acquired knowledge from past experience.

In this context, the EU-FP7 European initiative e-LICO[10] proposed the DMOP ontology (Keet et al., 2015), which is defined to support the analytic workflow composition by following the standard CRISP-DM (Shearer, 2000). DMOP is used to define analytical workflows, as well as to describe algorithms, parameters, inputs/outputs and a large amount of meta-data included in typical data mining processes. A step further was taken by Kumara, Paik, Zhang, Siriweera, and Koswatte (2015) that use Automatic Service Composition to automate the analytic workflow generation.

As a summary, Table 2 outlines the main features of the related work with regards to the semantic approach proposed here. These features consist of specifying whether the existing approaches: focus on data mining or optimization, are oriented to Big Data, provide proof-of-concepts, align with other ontologies, use OWL/RDF in the semantic model and/or describe workflow composition tasks. Then, it is possible to identify the actual contributions of the proposed semantic model beyond the state of the art, as follows:

- BIGOWL is conceived to semantically model data analytics in Big Data environments. Similarly to other ontologies in the literature, it is oriented to general KDD procedures, although considering those Big Data ecosystem elements with class instances, e.g., ontology individuals.
- It is aligned with the DMOP ontology, which is in turn aligned with CRISP-DM. They have been validated to construct data mining workflows.
- Besides data mining, BIGOWL is also focused on optimization algorithms, although with special interest on covering multi-objective metaheuristics in Big Data environments.
- The proposed approach is validated on two real-world use-cases consisting of classical data mining and streaming data processing for multi-objective optimization.

3. Current practices in Big Data analytics

In current Big Data technology ecosystems, when facing a specific data analytic task, it is usual to support on already existing tools. Some of those consist in commercial services often provided through cloud computing Software-as-a-Service (SaaS), which can be used by no skilled people by means of workflow compositions (e.g., Azure ML, Amazon ML, BigML, Data Mining Cloud Framework, and Kognitio); other tools are open-source frameworks requiring skilled users who prefer to program their application using more technical approaches. Additional factors (such as: data format, data source, volume and velocity required to analyse data) are also determinant when choosing the proper technology (Zomaya & Sakr, 2017). Hadoop ecosystem represents the most used framework for developing distributed Big Data analytic applications. However, it is conceived for high skilled users, so even the standard workflow composition service of Hadoop (Oozie) requires certain programming ability to be properly used.

Besides technological or commercial aspects, current Big Data platforms still follow the common procedure when facing data analytics tasks (ACM-SIGKDD, 2014), which comprises typical steps of classical KDD: data collection, data transformation, data mining, pattern evaluation, and knowledge presentation.

Keeping this in mind, the proposed semantic approach is oriented to general KDD procedures, then leading the underlying Big Data technological platform to be semantically annotated with class instances, e.g., individuals in the ontology.

4. Semantic model

One of the main goals in this study is to capture all the needed semantics to guide the smart design of Big Data analytics workflows and to enhance their performance. For this reason, we opted to design an OWL 2 ontology to describe analytic algorithms, datasets, problems, and workflows in the Big Data context.

To this end, the standard Ontology 101 development process (Noy & McGuinness, 2001) has been followed, which comprises seven steps:

1. *Determine the domain and scope of the ontology*. The main scope of BIGOWL is data processing and data analytics in Big Data en-

[10] http://www.e-lico.eu/.

Table 2
Summary ontologies' features.

Feature/Ontology	CRISP-DM	KDDONTO	PMOEA	ECO	(Pinto'2015)	(Phan'2015)	DMWF	KD	DMOP	BIGOWL
Data Mining	✓	✓			✓	✓	✓	✓	✓	✓
Optimization			✓	✓						✓
Big Data environments										✓
Proof of concepts	✓	✓	✓	✓			✓	✓	✓	✓
Aligned to other ontology									✓	✓
OWL/RDF	✓	✓	✓	✓			✓	✓	✓	✓
Workflow composition							✓	✓	✓	✓

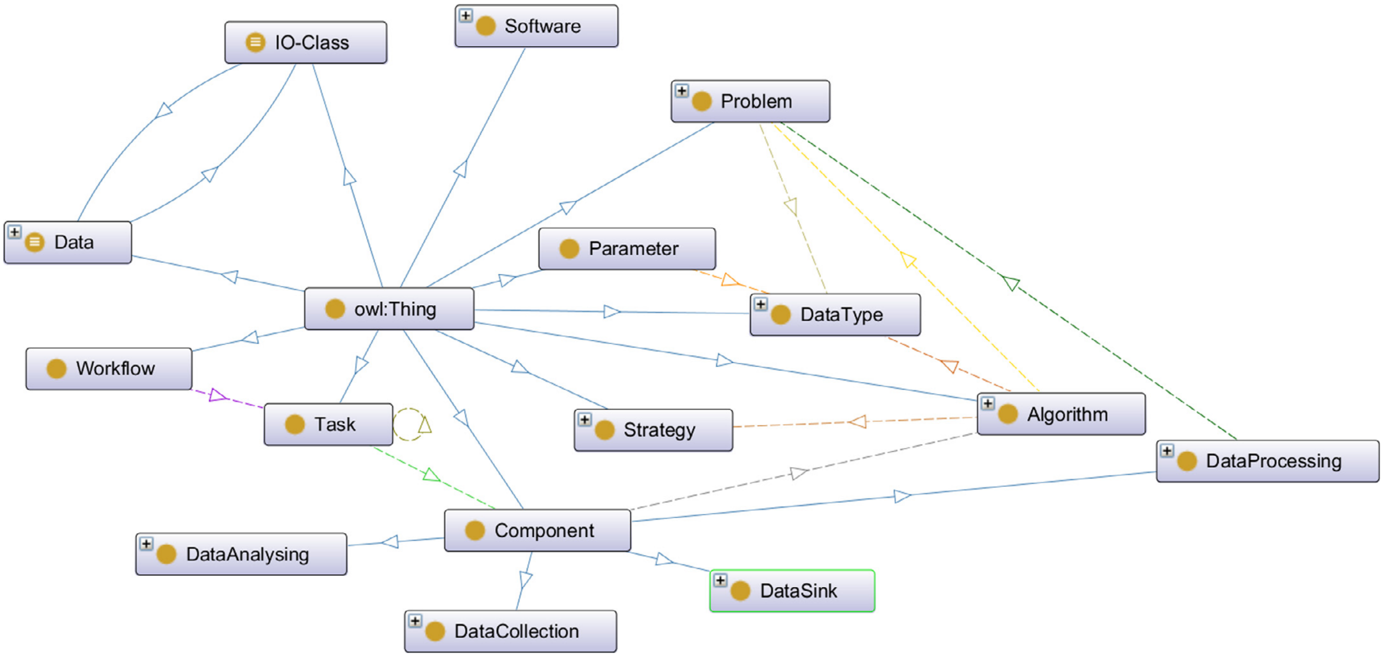

Fig. 1. Overview of the BIGOWL ontology. Continuous arrows refer to subclasses, whereas dotted ones refer to properties.

vironments. This considers not only classical data analytic procedures, but also specific data processing and underlying software platform features oriented to Big Data.

2. *Consider reusing existing ontologies*. As commented before, the proposed ontology is aligned with DMOP, which has been successfully validated to construct data mining workflows. DMOP is in turn aligned with the foundational ontology DOLCE (Masolo, Borgo, Gangemi, Guarino, & Oltramari, 2003) and follows the standard CRISP-DM in the definition of data mining processes.
3. *Enumerate important terms in the ontology*. Important terms were selected from the literature related to Big Data and optimization. In addition, terms from the ontologies aligned (Keet et al., 2015; Yaman et al., 2017) were also incorporated. Examples of such terms are: *Component, Workflow, Task, Data, DataProcessing* and *Software*.
4. *Define the classes and the class hierarchy*. We have followed a top-down approach in developing the class hierarchy. This fact facilitates among others, the alignment with DMOP and DOLCE, the design of annotation mappings and the use of a semantic reasoner. Fig. 1 shows the ontology core classes and hierarchy. For instance, the class *Component* has several subclasses, including *DataAnalysing* and *DataCollection*. Classes modeling algorithms, components and workflows are aligned with the class *dmop:DataType*. BIGOWL has been developed using Protégé[11] and OWL 2.
5. *Define the properties of classes and slots*. With the purpose of relating classes and defining attributes, we have included object and data properties. A representative set of properties are shown in Table 3, where the class *Component* is related to class Algorithm by means of the object property *hasAlgorithm*. Data properties of class *Component* are *path, author, numberOfInputs* and *numberOfOutputs*.
6. *Define the facets of the slots*. This step includes the definition of cardinality constraints and value restrictions for the ontology's properties. For example, the range of the property *order* is restricted to integer (to specify in which step this task is carried out), when the class *Task* is its domain.
7. *Create instances*. Instances or individuals in BIGOWL are specific of the Big Data analytics domain. For example,*GeneratorDataTraffic* is an instance of the class *Kafka*, which is a subclass of *DataIngestion*. The class *Kafka* has a property *topicKafka* (with range "string") to indicate streams of records of Apache Kafka[12] services.

4.1. The BIGOWL ontology

BIGOWL has been developed following the steps described above, producing 184 classes, 16 object properties (binary relationships between individuals), 20 data properties (individual attributes), 488 axioms, 66 individuals and growing. It is worth mentioning that classes *DM-DataClass* ≡ *DMDataClass* and *IO-*

[11] https://protege.stanford.edu/.

[12] Data Streaming Processing https://www.kafka.apache.org/.

Table 3
Component: object and data properties.

Object properties	Description logic
hasAlgorithm	$\exists$ hasAlgorithm.Thing $\sqsubseteq$ Component
hasParameter	$\exists$ hasParameter.Thing $\sqsubseteq$ Workflow $\sqcup$ Algorithm $\sqcup$ Component
isConnected	$\exists$ isConnected.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
isCorrect	$\exists$ isCorrect.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component
specifiesInputClass	$\exists$ specifiesInputClass.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
specifiesOutputClass	$\exists$ specifiesOutputClass.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
Data Properties	Description Logic
author	$\exists$ author.Datatype Literal $\sqsubseteq$ Workflow $\sqcup$ Algorithm $\sqcup$ Component $\sqcup$ Problem $\sqcup$ Software
hasDataValue	$\exists$ hasDataValue.Datatype Literal $\sqsubseteq$ DataType $\sqcup$ IO-Class $\sqcup$ Parameter $\sqcup$ Workflow $\sqcup$ Algorithm $\sqcup$ Component $\sqcup$ Problem
numberOfInputs	$\exists$ numberOfInputs.Datatype Literal $\sqsubseteq$ Algorithm $\sqcup$ Component
numberOfOutputs	$\exists$ numberOfOutputs.Datatype Literal $\sqsubseteq$ Algorithm $\sqcup$ Component
path	$\exists$ path.Datatype Literal $\sqsubseteq$ IO-Class $\sqcup$ Algorithm $\sqcup$ Component

Table 4
Task: object and data properties.

Object properties	Description logic
compatibleWith	$\exists$ compatibleWith.Thing $\sqsubseteq$ Task $\top \sqsubseteq \forall$ compatibleWith.Task
hasComponent	$\top \sqsubseteq \forall$ hasComponent.Component
isConnected	$\exists$ isConnected.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
specifiesInputClass	$\exists$ specifiesInputClass.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
specifiesOutputClass	$\exists$ specifiesOutputClass.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
Data Properties	Description Logic
order	$\exists$ order.Datatype Literal $\sqsubseteq$ Task $\top \sqsubseteq \forall$ order.Datatype

Class ≡ *Data* are declared as equivalent (with relation ≡) to align with those classes from other ontologies (DMOP) that describe similar concepts. We use OWL-DL syntax (see Table 1) to formalize the proposed ontology. The complete ontology is developed in "`bigowl.owl`" file and available in the GitHub repository.[13]

A representative set of the main classes are described here, together with their object and data properties. These classes are: *Component, Task, Algorithm, Data*, and *Workflow*. Each class has defined a set of properties or conditions in order to be conceptualized. That is, an individual that satisfies those properties is considered to be a member of that class.

- Component. This class represents each processing step in the analytic workflow. It is used to encapsulate one concrete functionality, its parameters and the corresponding inputs and outputs it considers. The class *Component* has four subclasses that are oriented to define specific functionalities in typical data analytics processing chains: *DataCollection*, to connect to data sources; *DataProcessing*, to clean, curate, fuse and consolidate data; *DataAnalysis*, to perform the algorithmic function; and *DataSink*, to represent final steps in the data flow, e.g., store and visualization. Table 3 contains the object and data properties defined for *Component*. In accordance with these, a component can specify Input classes and Output classes, to define the type of data it is accepting and generating, respectively. Therefore, a component can connect with other one if their linking inputs and outputs are compatible among them.

- Task. A task represents an instance of a component that is used in a workflow and can be run. As shown in Table 4, the class *Task* has similar properties to those of *Component*, but including the object property *compatibleWith*, to specify compatibility among connected tasks, and the data property *order*, which indicates the specific step of execution in which this task is scheduled, in the scope of the workflow. A Component is then a template for one or more tasks, which will be used to carry out its specific functionality in a workflow.

- Algorithm. This class is devoted to cover all possible kinds It has two main subclasses: *DataMiningAlgorithm* and *OptimizationAlgorithm*; which are used to distinguish between these two families of algorithms. The former one is included in form of equivalence with the class *DM-Algorithm*, which is linked from DMOP. This way, all subclasses deriving from this class in DMOP are also used in BIGOWL. For the later, i.e., *OptimizationAlgorithm*, a new hierarchical classification of classes has been elaborated in this study for the annotation of this family, which comprises: *Exact, Heuristic*, and *Metaheuristic* algorithms as main subclasses.

Table 5 includes the object and data properties of *Algorithm*. Among its main object properties it is worth mentioning: *implements*, which is referred to a learning model or search strategy; *manages*, to annotate the type of data it works; and *resolves*, which is related to the *Problem* it is oriented to solve. This is a useful mechanism to relate classes *Algorithm* and *Problem*, which also share the data property *dealWith* that indicates the specific features an algorithm should fulfill to deal with a problem.

In this regard, the class *Problem* defines a series of data properties like: *numberOfConstraints, numberOfObjectives, encodedBy*, and *numberOfVariables*, that will lead a future reasoner to recommend the correct algorithm to solve it. These two classes have to be declared as *DisjointWith*, in order to avoid future inconsistencies when querying the annotated data in a workflow.

- Data. The class *Data* is devoted to annotate all the data flowing throughout the analytic workflow. It is declared as *EquivalentTo IO-Class* of DMOP. This aligning enables datatypes defined by third parties' ontologies to be contextualized in the analysis. Table 6 contains the main data properties defined for this class, namely: *path*, to annotate the origin of data; and *hasDataType*, which defines the relation with class *DataType*. This last is used to define the type of data, i.e. *PrimitiveType* (Double, Integer, Boolean, etc.) or *StructuredType* (Graph, Tree, Matrix, Vector, Tuple, etc.).

- Workflow. It is used to guide the correct orchestration of those tasks involved in a data analysis job. Its main object properties are *hasTask* and *hasParameter*, which are formally described in Table 7. These properties are used by the workflow to obtain the execution order, as well as the input/output specifications of each

[13] URL link https://www.github.com/KhaosResearch/BIGOWL.

Table 5
Algorithm: object and data properties.

Object properties	Description logic
hasComponent	$\top \sqsubseteq \forall$ hasComponent.Component
hasParameter	$\exists$ hasParameter.Thing $\sqsubseteq$ Workflow $\sqcup$ Algorithm $\sqcup$ Component
specifiesInputClass	$\exists$ specifiesInputClass.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
specifiesOutputClass	$\exists$ specifiesOutputClass.Thing $\sqsubseteq$ Algorithm $\sqcup$ Component $\sqcup$ Task
implements	Transitive Property implements $\exists$ implements.Thing $\sqsubseteq$ Algorithm $\top \sqsubseteq \forall$ implements.Strategy
manages	$\exists$ manages.Thing $\sqsubseteq$ Algorithm $\top \sqsubseteq \forall$ manages.DataType
resolves	$\exists$ resolves.Thing $\sqsubseteq$ Algorithm $\top \sqsubseteq \forall$ resolves.Problem
Data Properties	**Description Logic**
author	$\exists$ author.Datatype Literal $\sqsubseteq$ Workflow $\sqcup$ Algorithm $\sqcup$ Component $\sqcup$ Problem $\sqcup$ Software
hasDataValue	$\exists$ hasDataValue.Datatype Literal $\sqsubseteq$ DataType $\sqcup$ IO-Class $\sqcup$ Parameter $\sqcup$ Workflow $\sqcup$ Algorithm $\sqcup$ Component $\sqcup$ Problem
numberOfInputs	$\exists$ numberOfInputs.Datatype Literal $\sqsubseteq$ Algorithm $\sqcup$ Component
numberOfOutputs	$\exists$ numberOfOutputs.Datatype Literal $\sqsubseteq$ Algorithm $\sqcup$ Component
dealWith	$\exists$ dealWith.Datatype Literal $\sqsubseteq$ Algorithm $\top \sqsubseteq \forall$ dealWith.Datatype

Table 6
Data: object and data properties.

Object properties	Description logic
hasDataType	$\exists$ hasDataType.Thing $\sqsubseteq$ Parameter $\sqcup$ Data $\top \sqsubseteq \forall$ hasDataType.DataType
path	$\exists$ path.Datatype Literal $\sqsubseteq$ IO-Class $\sqcup$ Algorithm $\sqcup$ Component

Table 7
Workflow: object and data properties.

Object properties	Description logic
hasTask	$\exists$ hasTask.Thing $\sqsubseteq$ Workflow $\top \sqsubseteq \forall$ hasTask.Task
hasParameter	$\exists$ hasParameter Thing $\sqsubseteq$ Workflow $\sqcup$ Algorithm $\sqcup$ Component
Data Properties	**Description Logic**
author	$\exists$ author.Datatype Literal $\sqsubseteq$ Workflow $\sqcup$ Algorithm $\sqcup$ Component $\sqcup$ Problem $\sqcup$ Software
hasDataValue	$\exists$ hasDataValue.Datatype Literal $\sqsubseteq$ DataType $\sqcup$ IO-Class $\sqcup$ Parameter $\sqcup$ Workflow $\sqcup$ Algorithm $\sqcup$ Component $\sqcup$ Problem
isCorrectWorkflow	$\exists$ isCorrectWorkflow.Datatype Literal $\sqsubseteq$ Workflow $\top \sqsubseteq \forall$ isCorrectWorkflow.Datatype
numTasks	$\exists$ numTask.Datatype $\sqsubseteq$ Workflow $\top \sqsubseteq \forall$ numTask.Datatype

task. This information, together with the data properties *numTasks* and *isCorrectWorkflow*, is then used in reasoning time to check whether the workflow is correctly composed or not, i.e., to address semantic validation of the analytic workflow.

4.2. Overall approach

An overview of the proposed semantic model is illustrated in Fig. 2, which is arranged together with the underlying operational model, hence enabling actual composition of analytic workflows.

In this approach, BIGOWL is the ontological scheme driving the whole process. It is the terminological box (TBox) that defines the vocabulary with concepts and properties in the domain of Big Data analysis. As explained before, BIGOWL is developed in OWL 2 according to which, concepts are represented by classes and relations are represented by data properties or object properties. As represented in Fig. 2, BIGOWL is conceived as an abstract top-level ontology that enables not only subontology replication e.g., to focus on specific use cases or algorithmic families, but also linkage with external domain knowledge ontologies, which are oriented to the specific problem domain (Smart Cities, Biology, etc.).

At bottom-level, the Assertional Box (ABox) defines all the instances in the knowledge domain (in OWL 2 an instance is represented by an individual) involving the analytic workflows' metadata. These instances are stored in RDF triple format in a Stardog[14] repository, which is a commercial version of the Pellet OWL 2 reasoner (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007), but enhanced with persistence capabilities. Once the ontology (Tbox) has been loaded together with SWRL rules, a series of reasoning tasks are launched by using the Stardog OWL 2 reasoner to derive new information that is not explicitly expressed in the knowledge base. The new information will indicate, when applicable and among others, whether an analytic workflow is correctly composed, or not.

In this model, the Annotation Module is used to populate the RDF repository with new instances that involve the required metadata (annotated) to be used in workflows, for example: algorithms, operators, parameters, input/output (paths), data sources, database connections, data sinks, software, execution order, etc.

The Operational Model will make use of these annotated metadata for driving the workflow composition. In this process, each step a new component is to be selected and used, a SPARQL query is launched to obtain the required meta-data and to suggest the next possible component/s to be included.

A very simple (hypothetical) case of use would comprise the following steps:

(i) A user desires to extract patterns from a dataset and visualize the results;
(ii) Then, the user selects one algorithm from a list of data mining algorithms (in form of analysis component) queried throughout the semantic model;
(iii) The selected algorithm requires specific input parameters and data to train, so the semantic model will supply them;
(iv) The initial dataset should be then formatted in form of data collection task;
(v) In case collected data need transformation, an intermediate data processing component is included between collection and analysis;

[14] http://www.stardog.com/.

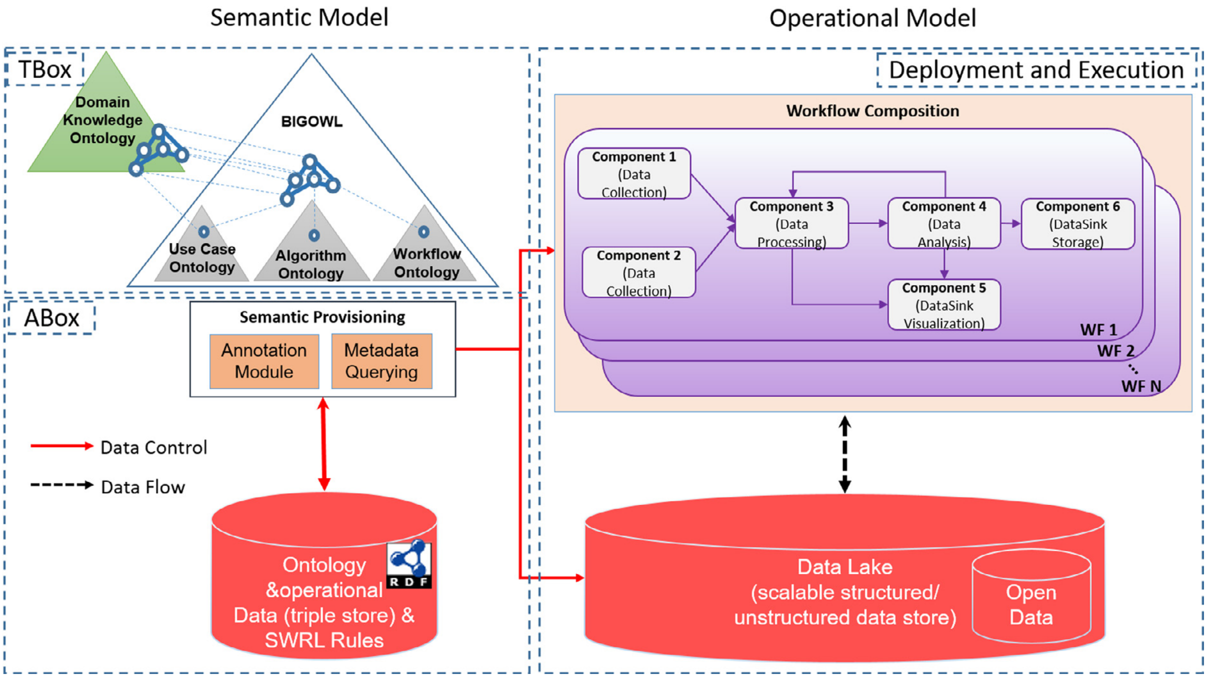
Semantic Model
Operational Model
TBox
Domain Knowledge Ontology
BIGOWL
Use Case Ontology
Algorithm Ontology
Workflow Ontology
ABox
Semantic Provisioning
Annotation Module
Metadata Querying
Data Control
Data Flow
Ontology &operational Data (triple store) & SWRL Rules
RDF
Deployment and Execution
Workflow Composition
Component 1 (Data Collection)
Component 2 (Data Collection)
Component 3 (Data Processing)
Component 4 (Data Analysis)
Component 5 (DataSink Visualization)
Component 6 (DataSink Storage)
WF 1
WF 2
WF N
Data Lake (scalable structured/ unstructured data store)
Open Data

Fig. 2. General overview of the semantic model that follows the ontology's scheme of BIGOWL. The analytic operational model address the workflow composition driven by the semantic model

(vi) The semantic model will suggest suitable output component (visualization) to be linked after the analytic algorithm.

It is worth mentioning that each step in the workflow is instantiated by a task, which entails an execution order. Then, the entire workflow is arranged according to all the ordering values in tasks.

In summary, the semantic model acts as a mediator between data provider components and data consumers. It also acts as a data source and meta-data registry with functions to make "agreements" on the provision and traceability of the whole data value chain.

5. Validation

For validation purposes, two different cases of study have been developed to show how the proposed semantic approach is used for driving the composition of data analytic workflows. The first one is focused on Big Data streaming processing and optimization of real-world traffic routes in the domain of Smart Cities. The second case study is centered on classic data mining analysis on academic problem instances, although considering local and cloud computing environments. In this way, we aim at covering, as much as possible, different aspects in Big Data applications: algorithmic analyses (optimization and data mining), velocity and volume issues (streaming processing), real-world and academic data problems, and Big Data ecosystems (Apache Spark local and on-premise cluster, BigML cloud SaaS API).

In these two cases, a similar semantic annotation and querying procedure has been followed, which consists in the manual annotation (guided by domain experts) of: algorithms, technological/platform features, and attributes of problem domain of knowledge; and automatic querying by means of SPARQL sentences. To distinguish individuals belonging to each case study, two different namespaces has been defined, i.e. traffic: http://www.khaos.uma.es/perception/traffic/khaosteam# and weka: http://www.khaos.uma.es/perception/weka/khaosteam# , respectively.

5.1. Case study 1: streaming processing of New York City traffic open-data

The first case study consists in a dynamic version of the bi-objective Traveling Salesman Problem (TSP), to minimize the "travel time" and the "distance" to cover certain routing points in a urban area. The algorithm for solving it is a dynamic variant of the well-known multi-objective metaheuristic NSGA-II provided in jMetalSP (Barba-González, García-Nieto, Nebro, Cordero, Durillo, Navas-Delgado, & Aldana-Montes, 2017),[15] which allows parallel processing of evaluation functions in Apache Spark environment.

In the case of the dynamic bi-objective TSP, which is formulated in terms of a distance matrix and a time travel matrix, the periodic changes can affect any of them. Our particular dynamic TSP problem instance is based on real-world data. Specifically, it is feed from the Open Data API provided by the New York City Department of Transportation,[16] which updates traffic information several times per minute. The information is provided as a text file where each line includes the average speed to traverse the two end points defining a link in the most recent interval. The goal is then, given a list of nodes in New York city and the distances between each pair of nodes, calculate the shortest possible route that visits each node.

New York's traffic data is read periodically by an external application that writes a file in HDFS whenever new data are acquired, so we have implemented a streaming data component for that purpose. This component reads periodically the new data appeared in the specific directory (this is done automatically by Spark) and makes a simple processing: if a change in a link is detected (time or distance), then the corresponding problem matrices are updated.

The analysis of the streaming data sources can be carried out in parallel by using Spark. In fact, we used a Hadoop cluster com-

[15] https://www.github.com/jMetal/jMetalSP.

[16] https://www.data.cityofnewyork.us/Transportation/Real-Time-Traffic-Speed-Data/xsat-x5sa.

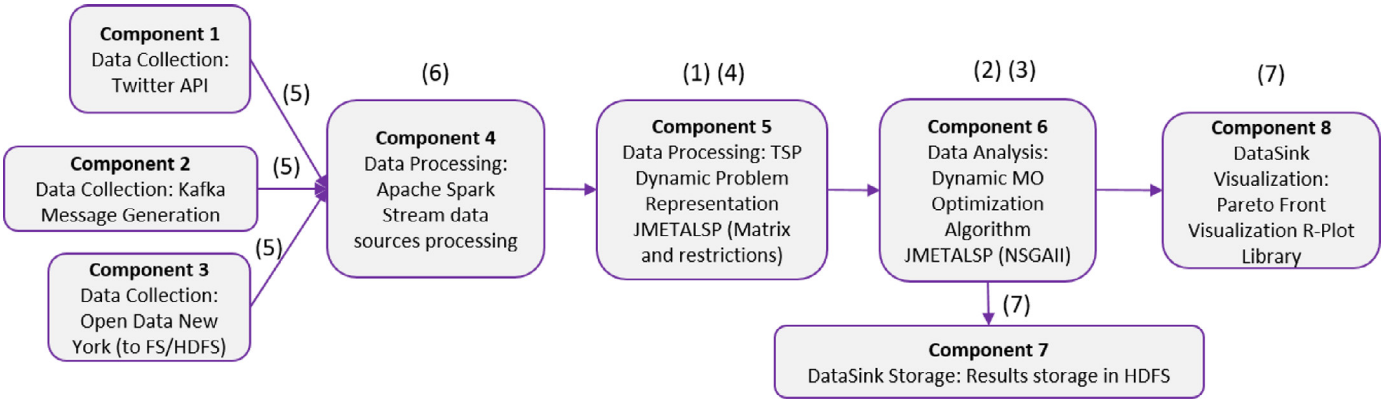
Component 1
Data Collection: Twitter API
Component 2
Data Collection: Kafka Message Generation
Component 3
Data Collection: Open Data New York (to FS/HDFS)
(5)
(5)
(5)
(6)
Component 4
Data Processing: Apache Spark Stream data sources processing
(1) (4)
Component 5
Data Processing: TSP Dynamic Problem Representation JMETALSP (Matrix and restrictions)
(2) (3)
Component 6
Data Analysis: Dynamic MO Optimization Algorithm JMETALSP (NSGAII)
(7)
Component 8
DataSink Visualization: Pareto Front Visualization R-Plot Library
(7)
Component 7
DataSink Storage: Results storage in HDFS

Fig. 3. Workflow for dynamic bi-objective optimization of TSP problem instance with Open Data New York

posed of 100 cores in the previous study where the Big Data optimization model was presented (Barba-González et al., 2017). In addition, two other streaming data sources where used as separate components, which based on Twitter and Kafka. In the first one, tweets are read from Twitter API with the topic "New York traffic" and a processing of each tweet is simulated, so the problem is updated in accordance with it (for testing purposes we set random changes in traffic scenario). This way, we combine a different streaming source with the possibility of adjusting the processing time, which will serve for performance evaluation purposes. In the second source, the idea is to enrich the case study with another data source that will produce artificial data. Then we created a Kafka message producer that generates, following uniform and normal distributions, a series of random messages with data to update the problem. Every 5 s at least 1000 messages are produced, but on average about 10,000 messages are created. Both the Twitter and Kafka streaming source classes have the same behavior as the HDFS based one: they iteratively collect and analyze the data to somehow update the problem.

After data processing, the analytic task is then carried out, which entails dynamic optimization computed by NSGAII algorithm of the jMetalSP library. The results of the analysis are used to feed data sinks. In this case study, we consider two of them: one that stores the produced Pareto fronts in HDFS, and other one that visualizes information about the Pareto front approximation (as the number of solutions and the number of generated fronts) using R-plot library.

The workflow implementing this case study is represented in Fig. 3,[17] where all the components are arranged according to data flow. In this workflow, the numeric indexes (1)–(7) correspond to those steps as indicated in Table 8, which contain the required SPARQL queries the semantic model apply to recommend forthcoming component/s to use, in design time. For this case study, the main set of individuals annotated in the semantic model and their relationships, are shown in Fig. 4. Then it is possible to follow the complete process step-by-step:

- **Step (1)**. The workflow designer fetch all the optimization problems from BIGOWL to select the implementation that better fits the required model for TSP instances. Interestingly, they are all subclasses of *OptimizationProblem*, which is integrated from DMOP. As a result, (s)he selects TSP.
- **Step (2)**. Given a problem to solve, TSP in this case, the semantic model recommends a series of optimization algorithms that could deal with it, i.e., those annotated algorithms that better adapt to the problem in terms of properties, such as: *solution encoding, manages, dealWith*, etc. After this, the designer selects NSGAII.
- **Step (3)**. This is an intermediate step followed by the semantic model to recommend specific annotated component and task instancing the underlying software that implements TSP and NSGAII.
- **Step (4)**. Now, the objective of this query is to obtain the specific data model to properly host data in problem and algorithm tasks. This step is thought to use specific domain knowledge information (traffic routes in this case) coming from external ontologies. The resulting annotated instance here is *MatrixNY*, which refers to a data model comprising a matrix of points and distances in the scenario of New York city.
- **Step (5)**. Once the workflow designer has a clear idea about the data model, (s)he can set data sources and connect them to feed the analysis. The semantic model is then queried to show all possible data collectors, i.e., those previously annotated. Among all the resulting possibilities, *ReadWebNYDataTraffic, DataCollectionDataTrafficKafka* and *DataCollectionTwitter* are selected for this case study.
- **Step (6)**. Before connecting data sources to analytic component, a previous task is required for data processing and consolidation. In this case study, the corresponding component is implemented as a Spark processing task to join Kafka messages, Tweets and traffic data streams.
- **Step (7)**. Last steps usually correspond to data sink tasks to allocate results from analyses. For this case study, *VisualizationTask* and *HDFSStoreTask* are selected, which implement R-plot visualization and storage in HDFS, respectively.
- **Step (8)**. Finally, the semantic model is queried to obtain the corresponding task instances that are mutually compatible among them. The analytic workflow is now ready to be launched on the underlying running platform.

Moreover, once the whole process is completed, a further reasoning procedure can now be started to check whether the generated workflow is semantically consistent, or not. This reasoning task will be explained in Section 5.3.

5.2. Case study 2: classification with Iris flower dataset

As commented before, the second case study consists in the academic problem of Irish flower classification by means of decision tree J48, a classical algorithm for data mining analytics. For materialization, two different approaches have been used in this case: the well-known library for data mining Weka and the BigML SaaS API for analysis on-cloud. The aim is to illustrate how similar annotation and querying procedures with BIGOWL can be used to

[17] Ontology instances available at https://www.github.com/KhaosResearch/BIGOWL/blob/master/traffic.owl.

Table 8
SPARQL queries for case study of streaming processing of New York city traffic open-data.

Step	SPARQL	Result

(1)	SELECT DISTINCT ?problem WHERE {
	 ?problem rdf:type ?type .
	 ?type rdfs:subClassOf* dmop:OptimizationProblem .}
		TSP, ZDT1, ZDT2, ZDT3, ZDT4, ZDT5, ZDT6, Kursawe..

(2)	SELECT DISTINCT ?algorithm (count(DISTINCT ?propertiesAlgorithm) AS numProperties) WHERE { traffic:TSP bigowl:encodedBy ?solution. ?algorithm rdf:type ?type. ?type rdfs:subClassOf* bigowl:OptimizationAlgorithm. ?entity bigowl:manages ?solution . ?algorithm bigowl:dealWith ?propertiesAlgorithm . traffic:TSP bigowl:hasFeature ?propertiesTSP . FILTER (?propertiesTSP in (?propertiesAlgorithm)). } GROUP BY ?algorithm ORDER BY DESC(?numProperties)	NSGAII, MOCell, SMSEMOA,SPEA2, IBEA, PAES, PESA2, WASFGA
(3)	SELECT distinct ?comp ?task WHERE { ?comp bigowl:hasProblem traffic:TSP . ?comp bigowl:hasAlgorithm traffic:NSGAII . ?comp rdf:type bigowl:Optimization . ?task rdf:type bigowl:Task . ?task bigowl:hasComponent ?comp. }	OptmimizationComponent, OptimizationTask
(4)	SELECT distinct ?data WHERE { ?comp bigowl:hasProblem traffic:TSP . ?comp bigowl:hasAlgorithm traffic:NSGAII . ?comp rdf:type bigowl:Optimization . ?task rdf:type bigowl:Task . ?task bigowl:hasComponent ?comp. ?task bigowl:specifiesInputClass ?data . }	MatrixNY

(5)	SELECT distinct ?dataCollection WHERE {
	 ?dataCollection rdf:type ?type.
	 ?type rdfs:subClassOf* bigowl:DataCollection.}
		ReadWebNYDataTraffic,
		DataCollectionHDFS,
		DataCollectionDataTrafficKafka,
		DataCollectionTwitter,
		DataCollectionDB, ...

(6)	SELECT distinct ?taskProcessing ?compProcessing WHERE { ?taskCollection bigowl:hasComponent bigowl:ReadNYDataTraffic. ?taskCollection bigowl:specifiesOutputClass ?out. ?dataProcessing rdf:type ?typeProcessing . ?typeProcessing rdfs:subClassOf* bigowl:DataProcessing. ?taskProcessing bigowl:hasComponent ?dataProcessing . ?taskProcessing bigowl:specifiesInputClass ?out. ?taskProcessing bigowl:specifiesOutputClass traffic:MatrixNY. }	SparkTask, ComponentSpark

(7)	SELECT distinct ?dataSink WHERE {
	 ?dataSink rdf:type ?type.
	 ?type rdfs:subClassOf* bigowl:DataSink.}
		VisualizationPlot,
		DataSinkHDFSStore,
		DataSinkOracleStore, ...

(8)	SELECT distinct ?task1 ?task2 WHERE {
	 ?task1 rdf:type bigowl:Task . ?task2 rdf:type bigowl:Task .
	 ?task1 bigowl:specifiesOutputClass ?output .
	 ?task2 bigowl:specifiesInputClass ?output . }
		GeneratorDataTrafficTask,
		SparkTask, TwitterCollectorTask,
		KafkaMGTask,
		ReadNYDataTrafficTask,
		OptimizationTask, VisualizationTask

compose workflows on different platforms when solving the same problem.

Fig. 5 shows the individuals (and their relationships) annotated in the ontology, and Fig. 6[18] represents graphically the analytic workflow for this case study. The numeric labels (1)–(5) are aligned with their corresponding steps in Table 9 that contain the SPARQL queries used and their results.

In a nutshell, steps (1)–(3) are used to guide the workflow designer on the selection of data model, algorithm, and analysis components and tasks, respectively. Step (4) is used to query suitable data collector components, in this case the designer selects *DataCollectionBigML* for BigML API instance and *DataCollectorFS* for Weka instance dataset. Step (5) queries are devoted to select possible data sink components, and specifically *DataSinkFSStore* and *Vi-*

[18] Ontology instances available at https://www.github.com/KhaosResearch/BIGOWL/blob/master/weka.owl.

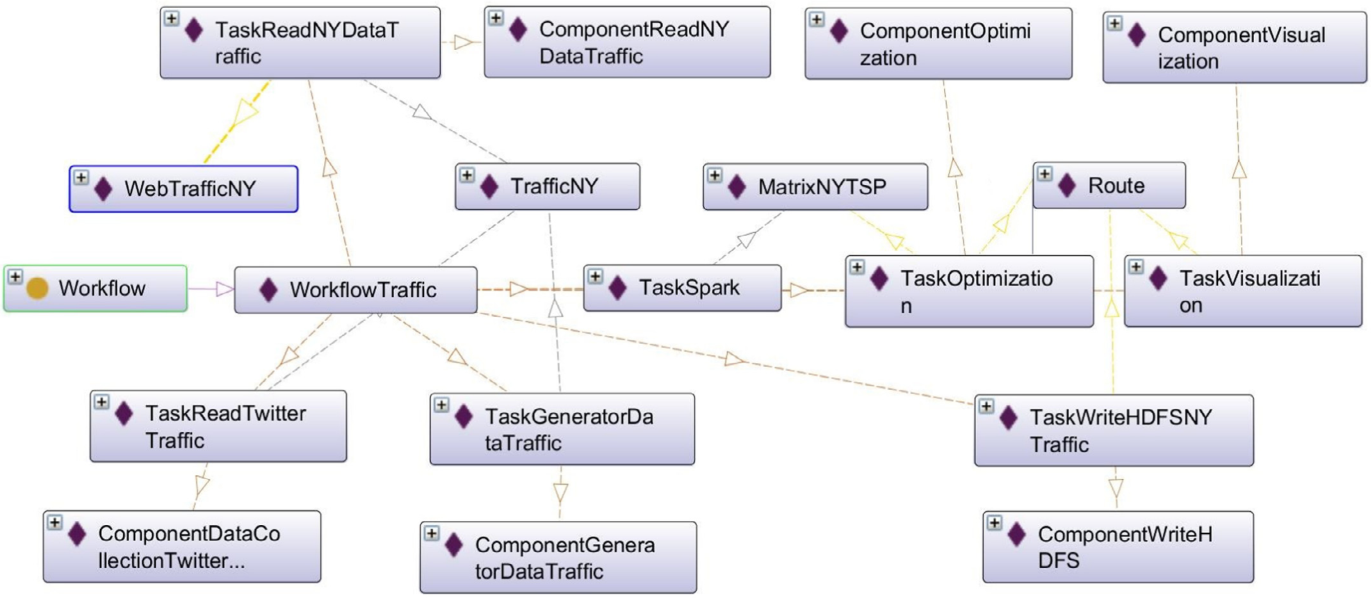

Fig. 4. BIGOWL's individuals annotated in the workflow for dynamic bi-objective optimization of TSP problem

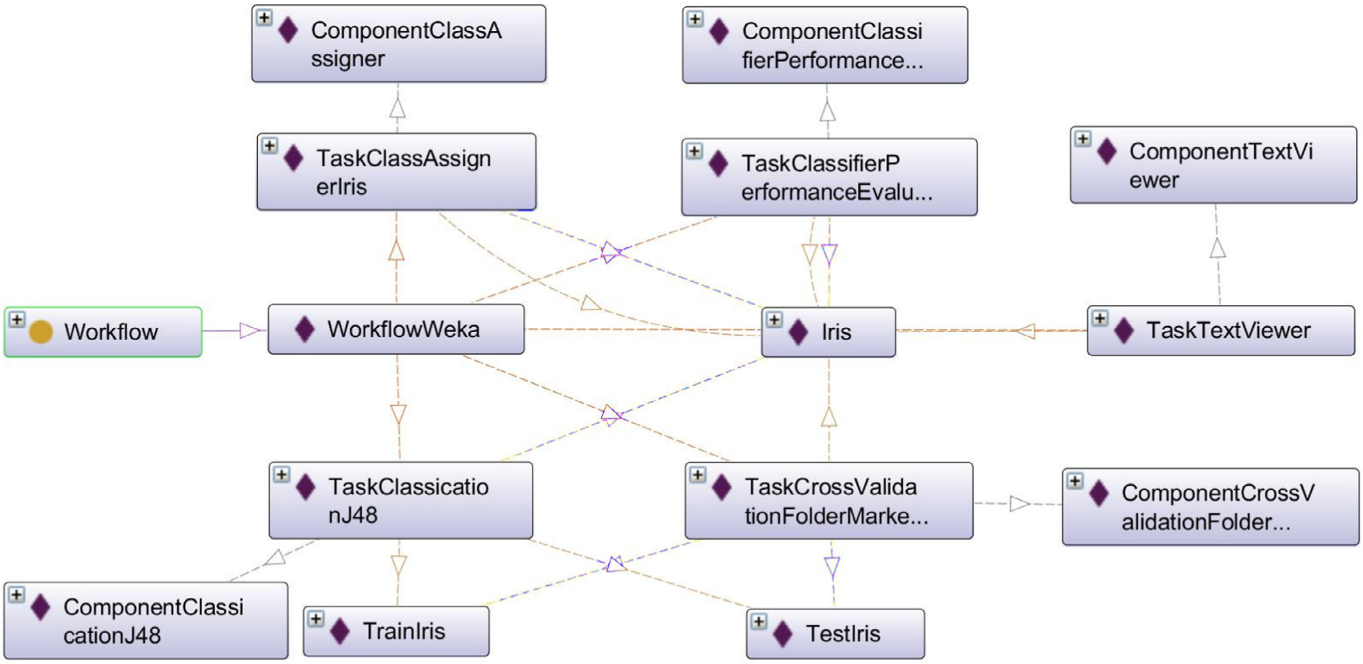
ComponentClassAssigner
ComponentClassifierPerformance...
TaskClassAssignerIris
TaskClassifierPerformanceEvalu...
ComponentTextViewer
Workflow
WorkflowWeka
Iris
TaskTextViewer
TaskClassicationJ48
TaskCrossValidationFolderMarke...
ComponentCrossValidationFolder...
ComponentClassificationJ48
TrainIris
TestIris

Fig. 5. BIGOWL's individuals in workflow for Irish flower classification with J48 decision tree instanced from Weka

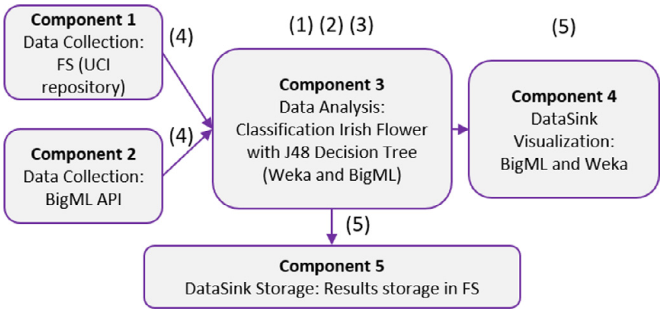
Component 1
Data Collection: FS (UCI repository)
(4)
Component 2
Data Collection: BigML API
(4)
(1) (2) (3)
Component 3
Data Analysis: Classification Irish Flower with J48 Decision Tree (Weka and BigML)
(5)
Component 4
DataSink Visualization: BigML and Weka
(5)
Component 5
DataSink Storage: Results storage in FS

Fig. 6. Workflow for Irish flower classification with J48 decision tree instanced from Weka and BigML.

sualizationPlot, which implement orders to save results in file system and API method for plotting in BigML, respectively. Finally, step (6) obtains the corresponding task instances that are mutually compatible among them throughout the complete workflow.

5.3. Reasoning with BIGOWL

Reasoning procedure is built in BIGOWL with formulation of semantic rules on top of the OWL ontology, to deduce new information from the existing knowledge. These rules are formulated in SWRL and used to perform semantic reasoning jobs mainly devoted to check correctness of workflows, e.i., to discover those components and tasks with (non-)compatible connectivity of inputs/outputs, execution orders, data domains, data formats, data

Table 9
SPARQL queries for case study Irish flower classification on Weka, as well as on BigML.

Step	SPARQL	Result

(1)	SELECT DISTINCT ?individual
	WHERE {
	 ?individual rdf:type ?type .
	 ?type rdfs:subClassOf* bigowl:DMDataClass .
	}
		Iris, Contact-lens, CPU, Diabetes, Glass, Ionosphre, Labor, ReutersCorn, Segment,..

(2)	SELECT ?algorithm
	WHERE {
	 weka:Iris rdf:type ?typeD .
	 ?typeD rdfs:subClassOf* ?classSomePropertyAlgorithm.
	 ?algorithm rdf:type ?type.
	 ?type rdfs:subClassOf* bigowl:DataMiningAlgorithm.
	  bigowl:DataMiningAlgorithm rdfs:subClassOf* [
	  a owl:Restriction ;
	  owl:onProperty bigowl:manages ;
	  owl:someValuesFrom ?classSomePropertyAlgorithm] .
	}
		J48, LogisticRegression, NaiveBayes, RepTree, IBk, LinearNNSearch, SMO, ...

(3)	SELECT distinct ?comp ?task
	WHERE {
	 ?comp bigowl:hasAlgorithm weka:J48 .
	 ?task rdf:type bigowl:Task .
	 ?task bigowl:hasComponent ?comp. }
		ClassificationJ48Component, ClassificationJ48Task

(4)	SELECT distinct ?dataCollection WHERE {
	 ?dataCollection rdf:type ?type.
	 ?type rdfs:subClassOf* bigowl:DataCollection.}
		DataCollectionOpenData, DataCollectionBigML, DataCollectionHDFS, DataCollectorFS, ...

(5)	SELECT distinct ?dataSink WHERE {
	 ?dataSink rdf:type ?type.
	 ?type rdfs:subClassOf* bigowl:DataSink.}
		VisualizationPlot, DataSinkHDFSStore, DataSinkOracleStore, DataSinkFSStore, ...

(6)	SELECT distinct ?task1 ?task2 WHERE {
	 ?task1 rdf:type bigowl:Task . ?task2 rdf:type bigowl:Task .
	 ?task1 bigowl:specifiesOutputClass ?output .
	 ?task2 bigowl:specifiesInputClass ?output . }
		ClassAsignerIrisTask, ClassificationJ48Task, ClassifierPerformanceEvaluatorTask, CrossValidaionFolderMarkerTask, TextViewerTask

types, etc. SWRL rules are then evaluated by the reasoner after classifying Big Data components in accordance with axioms, as defined in Table 1. In concrete, there are two types of axioms associated with OWL-DL classes for reasoning, namely: *subClassOf*, which is used to define the necessary conditions for a class to be considered a member of a given OWL class; and *equivalentClass*, for annotating when two classes can be considered as equivalent, if they comply the conditions.

BIGOWL imports *subClassOf* axioms from DMOP to specify taxonomy classification of Data Mining contexts and their data. In this sense, subclasses are also the natural way of describing hierarchy of algorithmic families and versions in optimization analyses. For instance, Genetic Algorithms are subclasses of Evolutionary Algorithms and these in turn, are subclasses of Population Based Algorithms. This structural information is then considered in reasoning time for algorithm recommendation. The main axioms for subclass classification are defined in Table 10, which correspond to Data Mining and Optimization algorithmic families.

Furthermore, a series of specific SWRL rules are described for assessing the compatibility of components. As commented before, the main goal is to address the generation of well-formed Big Data workflows. A description of these rules is as follows:

- Compatibility between task, component and Data Mining algorithm. This rule is used to check that input data model is compatible with the task that is indeed an instance (or implementation) of a component. In this specific case, the used component refers to a Data Mining Algorithm to perform a specific analysis. In short, this rule is used by the reasoner to validate compatibility between data mining component and data source. The result is a predicate indicating that data “feeding” the component are compatible with the analytic algorithm, so a task can be launched to run it on the underlying platform.

```
bigowl:specifiesInputClass(?task, ?data) ^
bigowl:hasComponent(?task, ?comp) ^
bigowl:hasAlgorithm(?comp, ?alg) ^
bigowl:DataMiningAlgorithm(?alg) ^
bigowl:DMDataClass(?data)
-> bigowl:isCorrect(?alg, ?data)
```

Note that a similar rule is defined in the semantic model to consider optimization algorithms.

- Compatibility between tasks of a workflow. This rule is applied to a complete workflow. It is used to check that input/output data connections of each pair of consecutive tasks are “semanti-

Table 10
OWL axioms for algorithmic subclass classification.

Class	Classification rule

Optimization Algorithm	`OptimizationAlgorithm subClassOf`
	`((implements some OptimizationStrategy) and`
	`(resolves some OptimizationProblem)) or Algorithm`

DataMining Algorithm	`OptimizationAlgorithm subClassOf`
	`(manages some DMDataClass) or Algorithm`

Optimization Component	`Optimization subClassOf (hasAlgorithm only`
	`(OptimizationAlgorithm or MachineLearning))`

DataMining Component	`DataMining subClassOf (hasAlgorithm only`
	`(DataMiningAlgorithm or MachineLearning))`

cally" similar. The outcome is a new predicate indicating whether each two consecutive tasks are mutually compatible, or not.

```
Workflow(?w) ^
bigowl:hasTask(?w, ?task1) ^
bigowl:order(?task1, ?ord1) ^
bigowl:hasTask(?w, ?task2) ^
bigowl:order(?task2, ?ord2) ^
swrlb:add(?ord2, ?ord1, 1) ^
bigowl:specifiesInputClass(?task2, ?data)^
bigowl:specifiesOutputClass(?task1, ?data)
-> bigowl:compatibleWith(?task1, ?task2)
```

- Connectivity between tasks and data. Similarly to the previous one, this rule is used to indicate that two instances of tasks are properly linked, that is to say, it checks that the input data of `task2` are covered with the output data of `task1`, according to the execution order established in the workflow.

```
Workflow(?w) ^
bigowl:hasTask(?w, ?task1) ^
bigowl:order(?task1, ?ord1) ^
bigowl:hasTask(?w, ?task2) ^
bigowl:order(?task2, ?ord2) ^
swrlb:add(?ord2, ?ord1, 1) ^
bigowl:specifiesInputClass(?task2, ?data) ^
bigowl:specifiesOutputClass(?task1, ?data)
-> bigowl:isConnected(?task2, ?data)
```

- Workflow correctness. Finally, this rule validates that all the components, instanced by corresponding tasks and data sources, are correctly arranged and connected. The result is then a new predicate indicating whether the complete workflow is correct, or not.

```
Workflow(?w) ^
bigowl:hasTask(?w, ?task) ^
bigowl:numberOfInput(?task, ?nIn) ^
bigowl:isConnected(?task, ?data).
sqwrl:makeSet(?set, ?data) ^
sqwrl:groupBy(?set, ?task).
sqwrl:size(?cont, ?set) ^
swrlb:equal(?cont, ?nIn)
-> sqwrl:select(?cont, ?nIn, ?task) ^
bigowl:isCorrectWorkflow(?w, true)
```

In summary, these case studies are used as a "*proof of concept*" to somehow highlight that the proposed semantic model is able to support in the design of Big Data analytics. In this regard, BIGOWL enables automatic SPARQL querying for component recommendation, as well as reasoning procedures for workflow validation.

6. Discussions

One of the main research findings we claim with the design and implementation of BIGOWL is the ability to represent and consolidate knowledge involving Big Data analytics. This semantic approach allows us to annotate (i.e. to "semantize") all the meta-data flowing from multiple data sources, processing components and analytic algorithms. The meta-data are integrated following the BIGOWL structure and stored in an RDF repository.

On the one hand, the results obtained in the two case studies indicate that, driven by the ontological model, it is possible to progressively deliver component recommendations for the construction of Big Data analytics workflows. The resulting workflows are indeed enhanced with semantic knowledge that explicitly describes and registers the data lineage (data provenance in database systems), from sources to results. It also would enable to replay specific portions or inputs of the data flow for step-wise debugging or regenerating lost outputs. In the BIGOWL semantic model, data linage is mapped with RDF triples referring to records of the inputs, entities, systems, algorithms and processes that influence data of interest, hence providing a historical record of the data obtained (as results) and its origins (as sources).

Based on the analysis provided in the two cases studies, the user is able to identify the correct path the data follow and how they are modified to obtain added value, for a given domain of knowledge. For example, in the first case study, a series of data sources involving information about urban traffic in the city of New York (with geo-locations, travel times, densities, tweets, etc.) are semantically related (or linked) to the results obtained, in form of optimized routes in a problem characterization of the classical TSP. In this case study, the outputs are encoded in form of routes, where the travel time and the routing distance are optimized. This way, the resulting routes are linked to the traffic densities and the Twitter messages, so the data lineage is registered with semantic annotations.

Similarly, in the second case study, it is possible to connect prediction accuracies with classification algorithms, for the Irish flower database. In addition, the running experiences acquired when using different execution frameworks, e.g., in-house/in-cloud, are also annotated as results.

Another important finding lies in the possibility of using the semantic knowledge-base, now consolidated in the RDF repository, to perform reasoning tasks, hence to infer new knowledge. In this study, a series of SWRL rules are used to train the reasoner. In this study, a reasoner is used to evaluate a set of SWRL rules defined for the specific task of workflow validation. In this regard, the validation analysis performed by the reasoner required 644 ms for case study 1 and 673 ms for case study 2. Taking into account that we used the Stardog OWL 2 reasoner, the time spent in reasoning tasks is acceptable for workflow validation.

On the other hand, the main constraint of the proposed semantic model is that it needs a domain ontology to cover the problem knowledge domain. This domain ontology contains the specific concepts for a given case, so it can be reused in domains where previous efforts provided such model. However, if such ontology is not available, then its design is required. As explained in Section 4.1, the class *Data* in BIGOWL is used, not only to annotate all the data flowing in the analytic workflow, but also to allow alignment with third parties' ontologies covering the specific problem domain of knowledge. Additionally, the general ontology could miss concepts that would be needed in some cases and are not described in the current model. This constraint can be solved by proposing an extension, in form of new version release of BIGOWL, though a collaborative portal. In this sense, BIGOWL is publicly available at WebProtégé,[19] where any registered user can introduce changes. These changes will be reviewed in a regular basis to approve or reject them. The last stable version of the ontology will be provided in the project GitHub repository.[20]

In addition, a secondary constraint arises when a new workflow is generated or executed by a user, since a series of new annotations are required to store all the meta-data involved in the data analytic process, in form of RDF triples. This makes the RDF repository to increase significantly, which would promote, not only future reasoning procedures to infer new knowledge from these data, but also their connection with other Linked Data. In this sense, the efficient management of large RDF repositories has become a challenging task attracting many scholars to research (Zomaya & Sakr, 2017), which means a clear implication for academia.

In terms of practical implications, the proposed semantic model represents an initial demonstrator for the experimental piloting of Big Data frameworks enhanced with semantics. The objective is to obtain "Smart Data" and promote the data value chain in industry processes, which is a key challenge nowadays as reflected in the Strategic Research and Innovation Agenda of the Big Data Value Association (EU SRIA 4.0 BDVA).[21] Several industrial projects in this association, like BigDataEurope[22] and BigOceanData,[23] are focused on exploiting semantics in Big Data analytics, so they could partially take advantage of BIGOWL as reference ontological model.

7. Conclusions

In this work, an ontological approach called BIGOWL is proposed to provide a conceptual framework for the annotation of Big Data analytics. The proposed semantic model is materialized by means of an RDF repository, and programmatic querying and reasoning functions.

To test the initial hypothesis, two case studies have been developed, which consist in: (1) real-world streaming traffic data processing for route optimization in urban environment, and (2) academic data mining classification on local/on-cloud platforms. The experience on these cases revealed that BIGOWL approach is useful when integrating knowledge domain concerning a specific analytic problem. Consequently, the integrated knowledge is used for guiding the design of Big Data analytics workflows, by recommending next components to be linked, and supporting final validation.

It is worthy to declare that the proposed semantic model is currently populated with those annotated elements required to set the case studies reported in this work, although it can be feed with new instances regarding other Big Data workflows.

This motivates our future research agenda, which entails a first phase to provide automatic facilities for ontology population, hence to enrich the semantic approach; second, to provide new mechanisms to promote the use of contextual domain of knowledge in the generation of Big Data analytic solutions; and third, to generate new and heterogeneous use cases of analytics workflows that would led us to find and solve new possible deficiencies, as well as to enrich the knowledge base.

References

ACM-SIGKDD (2014). Data mining curriculum. ACM SIGKDD 2006-04-30. Retrieved 2014-01-27.

Allahyari, M., Kochut, K., & Janik, M. (2014). Ontology-based text classification into dynamically defined topics. In *2014 IEEE international conference on semantic computing* (pp. 273–278).

Barba-González, C., García-Nieto, J., Nebro, A. J., Cordero, J. A., Durillo, J. J., Navas-Delgado, I., et al. (2017). Jmetalsp: A framework for dynamic multi-objective big data optimization. *Applied Soft Computing*. In–Press–Online

Diamantini, C., Potena, D., & Storti, E.. Ontology-driven kdd process composition.

Dou, D., Wang, H., & Liu, H. (2015). Semantic data mining: A survey of ontology-based approaches. In *Semantic computing (icsc), 2015 ieee international conference on* (pp. 244–251). IEEE.

Grosof, B. N., & Poon, T. C. (2004). SweetDeal: Representing agent contracts with exceptions using semantic web rules, ontologies, and process descriptions. *International Journal of Electronic Commerce, 8*(4), 61–97.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies, 43*(5–6), 907–928.

Gruber, T. R., et al. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*(2), 199–220.

Harris, S., Seaborne, A., & Prud'hommeaux, E. (2013). Sparql 1.1 query language. *W3C Recommendation, 21*(10).

Horrocks, I., Patel-Schneider, P. F., Bechhofer, S., & Tsarkov, D. (2005). OWL rules: A proposal and prototype implementation. *Web Semantics: Science, Services and Agents on the World Wide Web, 3*(1), 23–40.

Jing, L., Ng, M., & Huang, J. (2010). Knowledge-based vector space model for text clustering. *Knowledge and Information Systems, 25*(1), 35–55.

Keet, C., Ławrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., & Palma, R. (2015). The data mining optimization ontology. *Web Semantics, 32*, 43–53.

Kietz, J., Serban, F., Bernstein, A., & Fischer, S. (2010). Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In *Proceedings- of the ecml/pkdd: 10* (pp. 1–12).

Konys, A. (2016). Ontology-based approaches to big data analytics. In *International multi-conference on advanced computer systems* (pp. 355–365).

Kuiler, E. W. (2014). From big data to knowledge: An ontological approach to big data analytics. *Review of Policy Research, 31*(4), 311–318.

Kumara, B. T. G. S., Paik, I., Zhang, J., Siriweera, T. H. A. S., & Koswatte, K. R. C. (2015). Ontology-based workflow generation for intelligent big data analytics. In *2015 ieee international conference on web services* (pp. 495–502).

Li, L., Yevseyeva, I., Basto-Fernandes, V., Trautmann, H., Jing, N., & Emmerich, M. (2017). Building and using an ontology of preference-based multiobjective evolutionary algorithms. In H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, & C. Grimme (Eds.), *Evolutionary multi-criterion optimization: 9th international conference, EMO 2017, Münster, Germany, March 19–22, 2017, proceedings* (pp. 406–421). Cham: Springer International Publishing.

Marinica, C., & Guillet, F. (2010). Knowledge-based interactive postmining of association rules using ontologies. *IEEE Transactions on Knowledge and Data Engineering, 22*(6), 784–797.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., & Oltramari, A. (2003). Wonderweb deliverable d18, ontology library (final). *ICT Project, 33052*.

McBride, B. (2004). The resource description framework (rdf) and its vocabulary description language rdfs. In *Handbook on ontologies* (pp. 51–65). Springer.

McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C Recommendation, 10*(10), 2004.

Nguyen, P., Hilario, M., & Kalousis, A. (2014). Using meta-mining to support data mining workflow planning and optimization. *Journal of Artificial Intelligence Research, 51*, 605–644.

Noy, N., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology. Technical report.*

[19] WebProtégé https://www.goo.gl/F6fYUc.

[20] GitHub https://www.github.com/KhaosResearch/BIGOWL.

[21] http://www.bdva.eu/sites/default/files/BDVA_SRIA_v4_Ed1.1.pdf.

[22] https://www.big-data-europe.eu/.

[23] http://www.bigoceandata.com/.

Noy, N. F., McGuinness, D. L. et al. (2001). Ontology development 101: A guide to creating your first ontology.

Phan, N., Dou, D., Wang, H., Kil, D., & Piniewski, B. (2015). Ontology-based deep learning for human behavior prediction in health social networks. In *Proceedings of the 6th ACM conference on bioinformatics, computational biology and health informatics* (pp. 433–442). ACM.

Pinto, A., Scioscia, F., Loseto, G., Ruta, M., Bove, E., & Sciascio, E. D. (2015). A semantic-based approach for machine learning data analysis. In *2015 IEEE international conference on semantic computing (ICSC)* (pp. 324–327).

Prud, E., & Seaborne, A. (2006). Sparql query language for rdf. *W3C Recommendation*.

Ristoski, P., & Paulheim, H. (2016). Semantic web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web, 36*, 1–22.

Roldán-García, M., García-Nieto, J., & Aldana-Montes, J. F. (2017). Enhancing semantic consistency in anti-fraud rule-based expert systems. *Expert Systems with Applications, 90*(Supplement C), 332–343.

Shearer, C. (2000). The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing, 5*(4), 13–22.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the WWW, 5*(2), 51–53.

Staab, S., & Studer, R. (2013). *Handbook on ontologies*. Springer Science & Business Media.

Yaman, A., Hallawa, A., Coler, M., & Iacca, G. (2017). Presenting the ECO: Evolutionary computation ontology. In *European conference on the applications of evolutionary computation* (pp. 603–619).

Záková, M., Kremen, P., Zelezny, F., & Lavrac, N. (2011). Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering, 8*(2), 253–264.

Zomaya, A. Y., & Sakr, S. (2017). *Handbook of big data technologies* (1st). Springer International Publishing.