Supporting Semantic Data Enrichment at Scale

Michele Ciavotta, Vincenzo Cutrona, Flavio De Paoli, Nikolay Nikolov, Matteo Palmonari, and Dumitru Roman

Abstract Data enrichment is a critical task in the data preparation process in which a dataset is extended with additional information from various sources to perform analyses or add meaningful context. Facilitating the enrichment process design for data workers and supporting its execution on large datasets are only supported to a limited extent by existing solutions. Harnessing semantics at scale can be a crucial factor in effectively addressing this challenge. This chapter presents a comprehensive approach covering both design- and run-time aspects of tabular data enrichment and discusses our experience in making this process scalable. We illustrate how data enrichment steps of a Big Data pipeline can be implemented via tabular transformations exploiting semantic table annotation methods and discuss techniques devised to support the enactment of the resulting process on large tabular datasets. Furthermore, we present results from experimental evaluations in which we tested the scalability and run-time efficiency of the proposed cloud-based approach, enriching massive datasets with promising performance.

Keywords Big data processing · Data integration · Data enrichment · Data extension · Linked data · Scalability

1 Introduction

Big Data and Business Analytics are among the main value-creating assets for private companies and public institutions—estimates indicate yearly earnings in the order of 274 billion dollars by 2022 [1]. This is made possible by theoretical

M. Ciavotta (✉) · V. Cutrona · F. De Paoli · M. Palmonari
Department of Informatics, Systems and Communication, University of Milan–Bicocca, Milan, Italy
e-mail: michele.ciavotta@unimib.it; vincenzo.cutrona@unimib.it; flavio.paoli@unimib.it; matteo.palmonari@unimib.it

N. Nikolov · D. Roman
Department of Software and Service Innovation, SINTEF AS, Oslo, Norway
e-mail: nikolay.nikolov@sintef.no; dumitru.roman@sintef.no

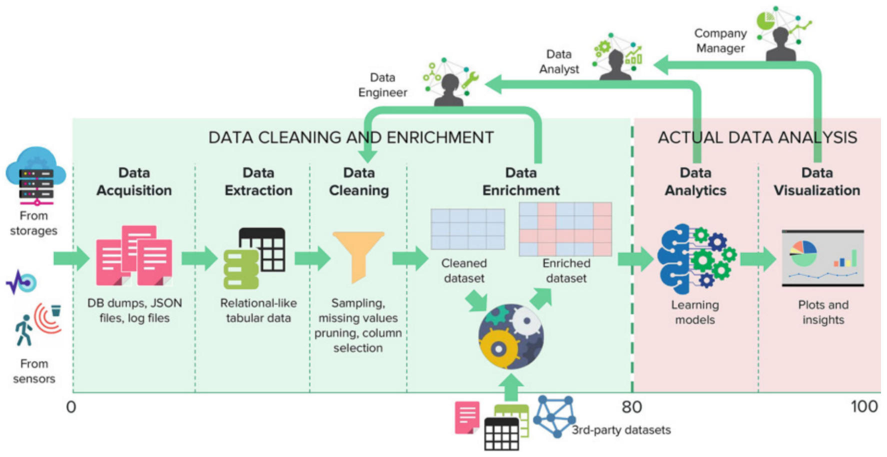


Fig. 1 Infographic representing the main stages of a data project and the related stakeholders

and practical advancements for processing massive amounts of data and developing highly accurate and effective decision-making processes via analytical models. However, very different time frame and effort are required to commission each phase of a data-driven project, which includes, as its primary stages, data acquisition, extraction, cleaning, integration/enrichment, and data analysis and results visualization [2]. Remarkably, the data preparation stage (which encompasses data transformations that also cover cleaning and integration/enrichment) takes up to 80% of the time required by a project. Only the remaining 20% of the time is spent on data analysis and exploration [3]. Such an imbalance (see Fig. 1) poses a problem that gets increasingly more severe with the progressive growth of volume and variability of the involved data [4]. This issue is now widely recognized and needs appropriate tools and methodologies, especially to support the crucial step of data enrichment.

Data enrichment is a specific data integration problem where a dataset that the user (typically a data engineer/scientist) knows is extended with additional information coming from external, possibly unknown, sources. Intuitively, enrichment requires *reconciliation* between values in the main and the external sources to fetch related data from the latter and extend the former. Data enrichment is often pivotal in analytics projects where the model might benefit from features that are not present in the main dataset, e.g., weather-based analysis of digital marketing campaign performance [5]. In recent years, a number of proposals have been presented, both academic and business related, to help data workers in the data preparation phase and, more specifically, in data enrichment tasks; many proposals involve the adoption of semantic techniques.

Semantics play an increasingly important role in Big Data, and, more specifically, in Big Data enrichment, as also acknowledged by the European Big Data Value Strategic Research and Innovation Agenda [2], and dedicated special issues in scientific journals [6–8]. A semantic paradigm that has gained popularity is based on Knowledge Graphs (KGs), which provide graph structures where entities are interconnected and classified. Semantic web technologies support the publication of KGs with standards and shared vocabularies that facilitate access and manipulation of knowledge via web protocols. Several approaches have been proposed to integrate data in different Knowledge Graphs, e.g., using entity reconciliation techniques [6]. However, in most data analytics problems, the user starts with some source legacy dataset that is not structured as a KG. Therefore, approaches have been proposed to transform legacy datasets by giving them a graph structure enriched with shared vocabularies, and, possibly, with background knowledge already available in a graph structure [9]. Such a transformation process is a complex task that can be (partly) sustained by semantic table interpretation [10] and annotation approaches. These approaches aim at mapping an initial relational table to the schema of a reference KG and finally linking values in the source table to entities in the KG [11, 12]. In this case, the focus of the interpretation algorithm is to automatically provide an annotation that enables the fusion of the source table with information in the target KG (e.g., a large and cross-domain information source like Wikidata), aiming at delivering an enriched KG. DAGOBAH [13] is an example of such an algorithm.

In this work, we leverage and take existing work on semantic table interpretation a step forward. We argue that semantic table annotation can provide a valuable paradigm to support data enrichment, modularly and at scale, in a much wider number of scenarios, including when the final objective is to enrich datasets, and not to their transformation into Knowledge Graphs. With *modularly*, we mean that the paradigm can be implemented by an ecosystem of services that provide access to different Knowledge Graphs to support automatic entity linking and data extensions. Automation is a key factor for managing large volumes of data and reaching the *at scale* dimension under certain assumptions that we discuss in this chapter.

We propose a comprehensive approach and a scalable solution to provide data workers with suitable tools to (1) interactively design transformation pipelines on datasets in tabular format, including semantic enrichment using curated knowledge bases (general purpose or domain specific), and (2) deploy and run such pipelines against massive datasets taking full advantage of the potential of scalability offered by modern Cloud services. Most of the related work in this field tackles the problem of automatically inferring the annotations that encode the semantics of a table. However, the primary contribution of this work consists of addressing the issue of implementing reconciliation and extension mechanisms to support both interactive data enrichment on small-size tabular datasets and automatic execution on massive workloads. To this end, we devised a two-phase approach and a service-oriented architecture to support it, whose engine consists of a collection of reconciliation and extension microservices implementing an open interface that can easily be scaled up to manage larger datasets.

To demonstrate the suitability of the proposed approach, we created both general-purpose services for linked data and specialized ones (for instance, for geographical toponyms, weather, and events), which support industry-driven analytic projects that motivated our work and guided a rigorous activity of requirements elicitation [14]. We used these services and business datasets to evaluate the efficiency of the proposed methods, achieving promising results (namely, linear scalability and a performance boost ranging from 4× to 770× over a baseline). Finally, we discuss the current limitations, which point to open issues in making semantic table enrichment approaches applicable at the Big Data scale. In this context, this chapter contributes to a better understanding of the role and challenges of semantics in supporting data enrichment, provides an approach and the corresponding implementation for semantic enrichment of tabular data at scale (thus, contributing to the *Data Management* and *Data Processing Architectures* horizontal concerns of the BDV Technical Reference Model [2] and to the *Knowledge and Learning* cross-sectorial technology enablers of the AI, Data and Robotics Strategic Research, Innovation and Deployment Agenda [15]), reports on the lessons learned in developing the solution, and presents the open problems for future research in the field of scalable semantic enrichment.

Ultimately, the proposed solution aims at filling an existing gap between technologies available today to support data enrichment at scale. This is a process where a natively semantic task like entity reconciliation plays a crucial role and semantics (especially KGs) are a facilitator of the enrichment process. Indeed, some tools provide users with user-friendly functionalities for data preparation, but few offer semantic support. The few solutions that offer such support (e.g., OpenRefine)[1] essentially cover the needs of the exploratory phases of a project by supporting manual transformation and enrichment of datasets. However, they neglect the life-cycle management needed to implement and run production-ready data pipelines and ensure scalability for large volumes of data. On the other hand, tools that provide support for running pipelines on large volumes of data are designed for users who are familiar with programming and process definition and are, therefore, unsuitable for use by data scientists [16]. Furthermore, these solutions, while often offering a wide variety of configurable components to create data pipelines, are poorly designed to incorporate user-specific knowledge, which is often essential to perform data reconciliation tasks effectively.

The rest of the chapter is structured as follows. Section 2 provides a discussion of the main design principles that have driven the definition of the architecture. The components of the platform and the workflow are discussed in Sect. 3. Section 4 illustrates experiments with datasets of different sizes. Finally, a review of the state of the art is reported in Sect. 5, and Sect. 6 concludes the chapter.

2 A Two-Phase Approach

Before discussing the proposed solution for data manipulation and enrichment at scale, we introduce a real-life analytics use case consisting of different data manipulation and enrichment tasks to motivate the principles that guided the formulation of the approach.

[1] http://openrefine.org

2.1 *Scenario: Weather-Based Digital Marketing Analytics*

The JOT Internet Media (JOT)[2] company is analyzing the performance of its digital marketing campaigns using reports from Google AdWords (GAW), and needs to aggregate data on performance (e.g., impressions and clicks) by city, region, or country. Furthermore, JOT, seeking to boost the effectiveness of future advertising campaigns, is interested in investigating the effect of weather on the performance of its campaigns at a regional level, and in training a machine learning model able to predict the most suitable moment to launch a campaign. To train the model and run the analytics, JOT aims to use 3 years of historical data concerning the performance of keywords used in previous campaigns.

The first step might be to enrich the GAW report (the white columns in Table 1) directly with weather data. In this scenario, JOT would access the European Centre for Medium-Range Weather Forecasts (ECMWF)[3] service that provides current forecasts, queryable using geographic bounding boxes and ISO 8601 formatted dates. Since both properties are missing in the original dataset, JOT has to first add them to the dataset. The ISO-formatted date can be easily obtained by applying a data transformation function to the date column (and adding the rightmost column in Table 1). Thus, the next step will be to extend the dataset with GeoNames (GN) identifiers for all mentioned locations. This operation requires to match the region labels adopted by GAW (Google GeoTargets labels) with GN identifiers, which are used in turn to geolocate the regions. Henceforth, this process is referred to as *data reconciliation* and represents a fundamental stage in the enrichment pipeline. Once the working dataset locations have been reconciled against GN (adding the fifth column in Table 1), it is possible to perform the *extension* step where the ECMWF is queried to collect the desired weather-related information (and add the sixth and seventh columns in Table 1). The reconciled and extended data are now suitable for performing the desired analysis.

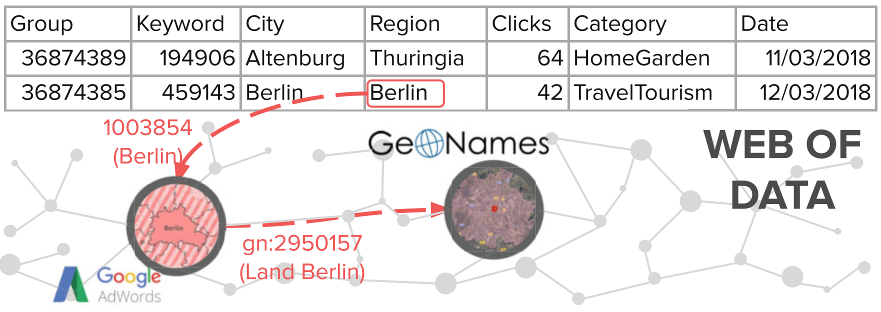
Table 1 JOT dataset enriched with data from GN and ECMWF

Keyword ID	Clicks	City	Region	Region ID (GN)	Temp. (WS)	Prec. (WS)	Date	Date (ISO)
194906	64	Altenburg	Thuringia	2822542	287.70	0.08	06/09/2017	2017-09-06
517827	50	Ingolstadt	Bavaria	2951839	288.18	0.02	06/09/2017	2017-09-06
459143	42	Berlin	Berlin	2950157	290.48	0.00	06/09/2017	2017-09-06
891139	36	Munich	Bavaria	2951839	288.18	0.02	06/09/2017	2017-09-06
459143	30	Nuremberg	Bavaria	2951839	288.18	0.02	06/09/2017	2017-09-06

Colored columns are appended by different functions: transformation (Date), reconciliation (Region ID), and extension (Temp., Prec.)

[2] https://www.jot-im.com

[3] https://www.ecmwf.int



Group	Keyword	City	Region	Clicks	Category	Date
36874389	194906	Altenburg	Thuringia	64	HomeGarden	11/03/2018
36874385	459143	Berlin	Berlin	42	TravelTourism	12/03/2018

Fig. 2 Semantics for data enrichment

2.2 *Semantics as the Enrichment Enabler*

As demonstrated by the above scenario, data enrichment plays a critical role in the preparation phase of many analytics pipelines, since it can add contextual information to the original dataset to build more effective models.

In the integration of relational datasets, traditional approaches (e.g., record linkage) have proven to be appropriate when the entity values of the involved schemas are compatible (e.g., they feature the same date format). In a more general scenario, however, the user is interested in enriching a dataset (known to the user) by fetching additional information from external datasets (possibly unknown to the user) that only relate semantically with the working dataset. This means that, in general, the terms of the schema and the values of the entities belong to different (possibly implicit) vocabularies or Knowledge Bases (KBs). The role of semantic approaches in such a process is to lift the latent semantics of records and metadata to support the integration of otherwise incompatible data sources.

In our use case, a semantic approach allows JOT to link the company dataset to the ECMWF data source by using a system of identifiers provided by a reference KB (i.e., GN). In this specific example, the reconciliation is performed directly against GN (see Fig. 2). In a more general case, it can happen that the data sources involved refer to different KBs; therefore, the KBs exploited for reconciliation need to be interlinked (e.g., using the *sameAs* predicate) to enable integration.

2.3 *Challenges*

Three main challenges emerge from the above scenario; indeed, the JOT data scientists have to: (1) investigate how to reconcile locations to GN, i.e., they need to look for a service that meets this requirement (suitable for users familiar with programming languages), or to check out the Knowledge Base (KB) that describes GN and build an ad hoc reconciliation service (suitable for users experienced in semantics and the geospatial domain); (2) query a ECMWF endpoint, i.e., they must look for the API documentation (usually geared towards users familiar with programming languages, less applicable to data scientists and domain experts); (3) come up with a scalable architectural solution able to manage and efficiently enrich the whole dataset, meeting possible time constraints. Specifically, since the enrichment process would unavoidably lead to querying external services, efficiency constraints require an effective solution to network latency issues that represent a bottleneck when a large number of API requests have to be issued.

In summary, the key features required to support the design and execution of enrichment-based data transformation at scale can be summarized as follows:

- Column values reconciliation against a reference KB, e.g., matching the spatial references adopted in the source dataset against the system of spatial identifiers adopted by the weather service.
- Data extension based on the reconciliation results, which represents the bridge between the dataset at hand and a reference KB. The extension could add one or more columns to the original dataset.
- An approach that supports the development of a user-friendly environment to design the reconciliation/extension process, and a scalable platform to execute it on massive input datasets.

2.4 Approach Overview

The approach we propose in this work is mainly based on a small-scale design/full-scale execution principle, harnessing semantics to support the reconciliation tasks in data enrichment. A high-level description of the approach is sketched in Fig. 3. The driving principle is to separate the transformation process into two phases: the *design phase*, where the user defines the transformation pipeline by working on a sample and produces a *transformation model* (i.e., an executable representation of the transformation pipeline), and the *processing phase*, where the model is executed against the original dataset to obtain an enriched version of it to feed the analytical activities. Both phases rely on external data sources (e.g., GN and ECMWF) to support reconciliation and extension activities.

A fully automated approach is unsuitable from a user perspective since it would entirely remove the operator control over the process and results. In processes where the matching phase is performed based on semantics, the contribution of the knowledge and experience of a domain expert can impact the final dataset quality. Therefore, complete automation would entail a substantial risk of generating low-quality results and, consequently, be of little use. For this reason, the approach gives full control over the definition of the transformations to the user while automating the enactment of the resulting process. The role of the human in the design phase is to define the pipeline steps over a smaller dataset so that she can control the resulting quality (e.g., they can avoid misinterpretations). In this scenario, the approach

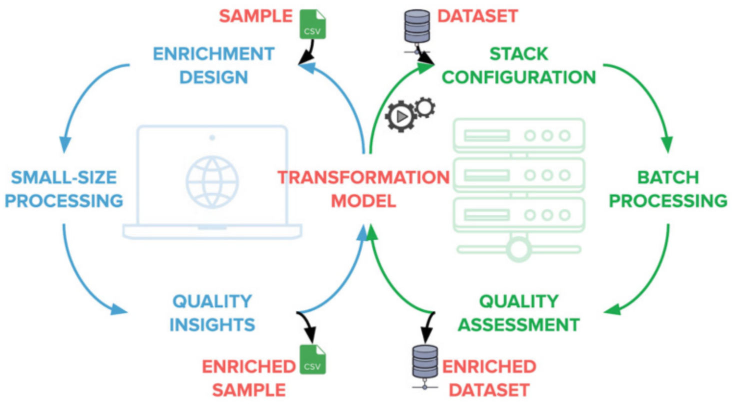


Fig. 3 Summary of design/processing approach

envisions computer-aided support tools offered as a service (the reader is referred to Sect. 3 for more details on the proposed solution) to guide the user in the pipeline composition, facilitating the process and reducing the need for strong programming skills. Once the transformation process has been defined, a full-scale *processing phase*, where the choices of the operator are packed within an executable artifact and run in batch mode over a different (possibly larger) dataset (Fig. 3), takes place.

As for the design phase, the approach proposes a reference implementation for the pipeline execution environment (see Sect. 3 for more details). Notice that featuring independent design and execution improves the efficiency of data management while mitigating confidentiality issues. Since the design phase exploits only a sample, the limited quantity of data to be transferred and manipulated simplifies the support system from an architectural and operational point of view (in practice, data would be handled by browsers on regular hardware) and reduces the risk of leakage of confidential information that might arise whenever full datasets are exposed. The full-scale execution phase can be performed on-premise, thus exploiting corporate infrastructure and tools, without the need for moving the data. Further details on the two phases are summarized as follows.

Design Phase In the design phase, the operator designing the pipeline performs three iterative steps on the working dataset: (1) the *enrichment design*, where the user designs each transformation step for enriching the working table employing a graphical interface that facilitates and automates interactions with reconciliation and extension services; (2) the pipeline execution (*small-size processing*), where each step of the enrichment process is performed over the current dataset; and (3) attaining *quality insights*, i.e., a handful of statistics to enable a general understanding of the overall quality of the result (e.g., the number of missing values). This interactive process is executed every time the user edits the pipeline definition (e.g., adding a new step in the pipeline). The outputs of the design phase are (1) the enriched dataset and (2) an executable transformation model, packaged in an executable that encompasses the steps of the pipeline. It is worth noting that if the user needs to enrich tables with a few thousand rows sporadically, the enriched table produced in the design phase concludes the process. If the user is required to enrich large volumes of data (or at least too large to be interactively managed), our approach assumes that the user carries out the design phase using a representative sample of the original dataset. In that case, the executable with the transformation model can be downloaded and used as the primary step in the processing phase (referred to as Data Flow), presented and discussed in the next section.

Processing Phase This phase aims to execute the transformation pipeline, which has been designed and tested on a smaller (loadable in memory) sample during the previous phase, on a large dataset. As in the previous phase, three steps are implied: (1) data flow definition (*stack configuration*) to support the execution of the enrichment pipeline; (2) *batch execution* (possibly in parallel) of the pipeline; and, finally, (3) *quality assessment* to evaluate the resulting dataset. If the result does not achieve an acceptable quality level (e.g., the number of reconciliation mismatches is above a given threshold), the user could go back to the design phase and modify the pipeline on an updated sample dataset. The new sample could be populated according to an analysis of the log files (e.g., adding a set of rows with values that could not be matched). The goal is to converge after a few iterations and be able to manage the dataset evolution. The *stack configuration* phase defines the pre- and post-processing actions to execute the enrichment pipeline on the dataset. It is composed of standard steps that can be customized according to the project requirements. To serve as an example, the reference data flow that supports the JOT scenario features the following steps: (1) decompress the input dataset and store it in a distributed file system; (2) split data in chunks for parallel processing; (3) execute the pipeline (which includes invoking enrichment services); (4) export the enriched data. The choice of relying on external services implementing the reconciliation and extension functionalities is supported by most available platforms and derives from precise design requirements (not least the requisites of modularity, extensibility, and flexibility). Consequently, the need to perform service invocations to get access to data for enrichment constitutes a fundamental scalability limitation of the entire enrichment process, which is much better performing and predictable for transformations that can be encapsulated within the executable transformation model. In the next section, the issue is discussed in detail.

Finally, on the one hand, these phases have different time requirements—i.e., in the design phase, the system has to be responsive in real time so that the user can provide feedback interactively, while the processing phase can last several hours. On the other hand, the phases deal with datasets of different sizes—the design phase processes only a sample of a dataset, while the processing phase must handle the full data. Therefore, we built an architecture where two main logical components share a set of back-end services to manage these phases. In the following, we provide details about the implementation of those components.

3 Achieving Semantic Enrichment of Tabular Data at Scale

This section outlines the different measures we took to ensure an adequate level of scalability for the enrichment process. The service architecture supporting the described approach is presented in Sect. 3.1. In Sect. 3.2 we discuss the design decisions, strategies, and lessons learned while designing for scalability in this domain, and examine the limitations of the proposed solution as well as possible improvements.

3.1 The Architectural View

The architecture underpinning the execution of our open-source solution, named ASIA (Assisted Semantic Interpretation and Annotation of tabular data) [17] (see Fig. 4), has been designed for modularity and loose coupling resulting in the components specified below:

ASIA User Interface The ASIA front-end is a single-page web application meant to interact with the final user exposing all the services required to support the enrichment. This application is fully integrated within Grafterizer [16] (part of DataGraft [18]), a tool that provides the pipeline abstraction, support for data cleaning and ETL data transformations, a tabular-to-Linked-data generator, and a compiler to produce portable and repeatable data manipulation pipelines. ASIA inherits and extends those features by providing functionalities to streamline the

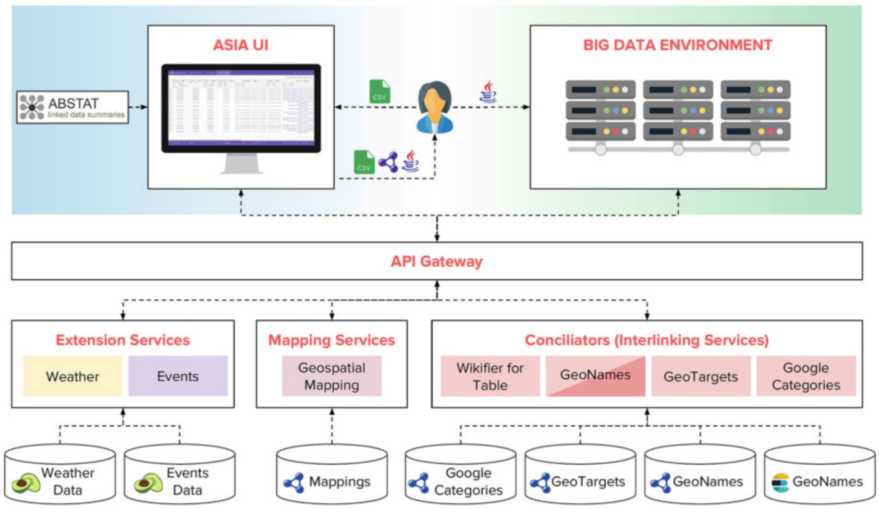


Fig. 4 Detailed architecture of ASIA

mapping between the columns of the working dataset and semantic types and properties (i.e., the schema annotation). This is done by analyzing the table headers and matching them with schema patterns and statistics provided by ABSTAT (Linked Data Summarization with ABstraction and STATistics) [19], an ontology-driven linked data profiling service. Moreover, ASIA UI provides widgets for semantic matching of column values against a shared system of identifiers.

ASIA Back-end This ecosystem consists of an orchestrator (API gateway) and a set of services[4] that are grouped in three categories: Conciliators, Mapping Services, and Extension Services.

- *API Gateway*. This service provides a unified view of the ASIA back-end ecosystem services by isolating the architectural details and relationships of these modules in the background. Moreover, it provides high-level functionalities by orchestrating the execution of the underlying services.
- *Conciliators*. Services for reconciling entity labels to a specific KB. They provide a REST interface compliant with the OpenRefine Reconciliation and Extension APIs.[5] In Fig. 4, conciliator blocks represent reconciliation services, while the GeoNames block also supports *KB-based extensions*, i.e., the possibility to extend the table with information from the reference KB.
- *Mapping Services*. Services in charge of linking KBs to each other, enabling the user to translate seamlessly between different Shared System of Identifiers (SSIs) by identifying URIs that provide suitable inputs to the extension services. The current implementation provides links between GeoTargets (location identifiers used in Google Analytics services) and GeoNames, plus additional *sameAs* links retrieved from DBpedia and Wikidata.
- *Extension Services*. Services for extending the input dataset with information coming from external data sources by using the URIs returned by conciliators and mapping services as inputs. In the JOT use case, we address weather and event datasets. The current implementation relies on data regularly being downloaded and curated (fetched from the ECMWF and EventRegistry[6] services, respectively). This is done to overcome network latency issues.

It is important to note that the ASIA back-end provides functionality to both the front-end (for the definition of the enrichment steps) and the Big Data Environment (presented below). The substantial difference is that in the first case its deployment is designed to provide multi-tenant support in Software-as-a-Service mode,[7] while in the second case, it is deployed in a dedicated way (and often replicating some services) to ensure scalability and efficiency.

[4] https://github.com/UNIMIBInside/asia-backend

[5] github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Developers

[6] eventregistry.org

[7] A DataGraft deployment that includes ASIA module is available online at https://datagraft.io

The Big Data Environment This macro-component is mainly responsible for the orchestration and execution of enrichment operations at scale. It provides a high-level interface to configure, process, and monitor data flows. In particular, to handle the scale of data, this component distributes the different replicas of the pipeline steps over a cluster of computational resources (physical machines). For this reason, the operator establishes an appropriate deployment in terms of resources and service replicas to support the parallel execution of each step and leaves the system with the burden of configuring, deploying, and running the flow. More details about this component can be found in the following section.

3.2 Achieving Scalability

The purpose of this section is to present the techniques and strategies employed to achieve a system capable of providing scalable enrichment functionalities.

Stateless, Shared-Nothing Processing The ecosystem of the ASIA back-end is made up of various services and databases capable of serving a number of concurrent invocations. In essence, ASIA services receive a label (in the case of reconciliation) or a URI (in the case of extension) and return one or more corresponding values. They are built to be stateless and thus enable the creation of a platform in which the enrichment pipeline is executed in parallel on non-overlapping segments of the working table (shared-nothing approach [20]).

Distribution and Parallelization The Big Data Environment (Fig. 5) is the component in charge of fostering parallelism and is implemented as a private cloud consisting of a cluster of bare-metal servers running the Docker engine,[8] connected via Gigabit Ethernet and sharing a distributed file system (i.e., GlusterFS).[9] In this environment, data flows are compiled into a chain of Docker containers that are, in turn, deployed and managed by a container Orchestration system (i.e., Rancher).[10] Each of the steps consists of containers working independently and in parallel and scalable on-demand. The communication between two consecutive steps of the chain, i.e., the handover of the partial results, occurs through writing and reading from the file system. For details on the approach for setting up the Big Data Environment, see [21]. The implementation of this container-based solution has several benefits: it makes the data flow deployment independent from the particular stakeholder's hardware infrastructure, also working in heterogeneous distributed environments; it guarantees a flexible deployment, better resource utilization, and seamless horizontal scalability. The GlusterFS distributed file system is fast (as it

[8] https://www.docker.com

[9] https://www.gluster.org

[10] https://rancher.com

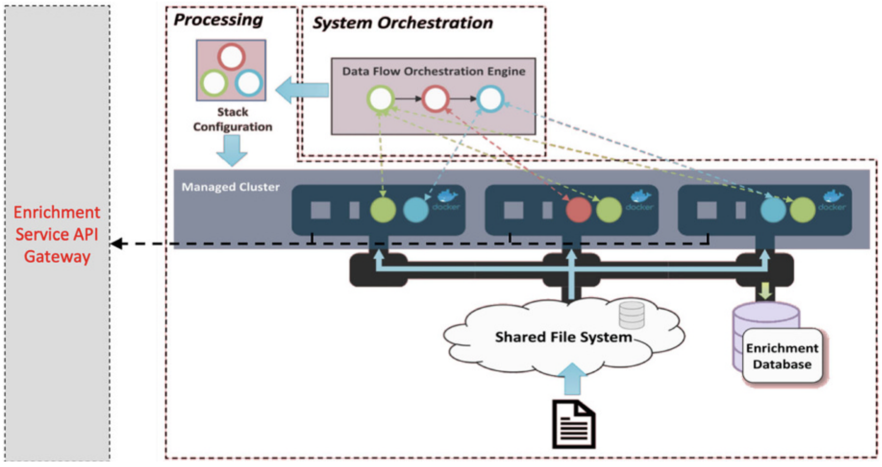


Fig. 5 An overview of the proposed Big Data Environment

lacks a central repository for metadata), linearly scalable, and, therefore, able to support massive amounts of data.

Data and Service Locality One of the primary issues to be addressed for achieving scalability for the enrichment process is the use of remote services. The use of services accessible over the Internet is certainly incompatible with datasets featuring more than a few thousand rows due to the network latency and the high number of invocations. The use of remote, multi-tenant services is generally acceptable in the design phase due to the limited size of managed datasets. However, when large datasets need to be processed, it is imperative to address the issue by making the life-cycle of enrichment data local. In the scenario we adopt, the weather information is downloaded daily from the provider and is treated to enable access using geospatial SSIs. This solution is suitable for datasets that change at a known rate and are thus stable. In the general case, refresh frequency depends on the application domain and the nature of data. The local management of these KBs has the advantage of rightsizing the resources allocated against the incoming workload; moreover, the control over the local network enables reduced and stable round-trip delay times (RDT). Similar considerations have led to deploying the reconciliation services of the ASIA back-end as close as possible (in network terms) to both the reference KBs and the agents (containers) performing the reconciliation pipeline steps.

Scaling up the Enrichment Services In order to manage the workload caused by the simultaneous invocation of reconciliation and extension functions (by the step executing the reconciliation pipeline), the ASIA ecosystem has been implemented to be easily replicable to achieve horizontal scalability of performance. This is achievable as the knowledge bases are used in read-only mode; accordingly, they can be duplicated without consistency issues. A load balancer is used for dispatching requests across the various replicas of ASIA.

Hierarchical Caching Lastly, it should be noted that the same request for reconciliation or extension can be made multiple times by the agents because columns in the processed data can contain repeating values. This, if not mitigated, generates a high number of identical requests. To address this, we implemented a hierarchical caching system in which each agent directly manages the first level of the hierarchy, while the other levels are managed by the stack of ASIA services and databases.

3.3 Discussion on the Limitations

In this section, we discuss the current limitations and aspects that could be improved to bring significant enhancements to the performance of the entire process.

Data Locality In this initial implementation of the Big Data Environment, the data locality principles, intended as one of the chief scalability enablers, are only partially implemented and exploited. Data locality is limited to the life-cycle management of the knowledge bases used by the enrichment services, which are physically brought to the Big Data Enrichment platform to reduce service access times. At the same time, the agents that perform the transformation pipeline may be physically separated from the working dataset they are processing (due to the use of a distributed file system). By deploying the enrichment services in the Big Data Enrichment platform, the speed of the functionalities relying on enrichment services increases dramatically. Similarly, the working dataset is stored in a distributed file system and accessible through the local network. This architectural choice, which enables uniform access times to data, has the disadvantage of raising the average read/write times of a quantity equal to twice the network latency (each agent reads a data chunk and writes a larger one). Nonetheless, by moving the data as close as possible to the agents that have to process it, we can improve the reading and writing performances and affect the whole process positively. This can be done by onboarding partitions of the working dataset on the machines that execute the containers of the agents instead of using the network to transmit partitions.

Distributed Caching The hierarchical caching system that was implemented can be further optimized, mainly because each ASIA replicated deployment has its local memory. Moreover, due to the presence of a load balancer running a round-robin dispatching policy (thus caching unaware), identical requests can be assigned to different replicas of ASIA causing preventable cache misses. The cache used at the agent level is also private, which results in generating much more requests than are strictly necessary. An improved solution to the problem of duplicated requests to the enrichment services can be done through the use of a distributed cache shared among the various instances of ASIA and among the agents that carry out the pipeline in parallel. Such a service (e.g., Ehcache [22]), once deployed on the machines that configure the cluster of the big data environment, would guarantee rapid synchronization of the local caches and would reduce the number of cache misses.

Efficient API Interaction The enrichment service API is another component that may be optimized to provide significant improvement to execution times. In the current implementation, for both the design and processing phases, reconciliation and extension are invoked for each row of the working table. This means that for each line the agent running the pipeline must wait a time equal to the RTD, forcing the system to wait a time roughly equal to twice the network latency at every invocation. A considerable improvement would be obtained by throttling the invocations to the service. The processing times of the input dataset could be further improved if light network protocols (such as Websocket [23]) were used together with improved message serialization (such as Google Protobuf [24]).

4 Evaluation of the Approach

To test the flexibility and scalability of the proposed solution, we performed three experiments of increasing scale involving real datasets. The experiments make use of two enrichment services: the geospatial reconciliation and extension service GN, and the weather extension service W, which is used to enrich the input dataset with weather information. GN takes only one attribute as input (e.g., a toponym), and creates one (reconciliation) or more (extension) columns; W takes two attributes as input—location and date—and appends as many columns as the number of desired weather features.

First, we designed a small-scale experiment reproducing the scenario where a data scientist executes the enrichment pipeline on a commodity machine (the whole cloud-native platform has been installed on a multi-tenant machine with 4 CPUs Intel Xeon Silver 4114 2.20 GHz, and 125GB RAM). The main objective was to assess the performance boost attributable to the introduction different caching levels. We started by testing the reconciliation performance with no caching strategy: 200 K rows (21 columns) from a real company dataset featuring 2227 different toponyms (from Germany and Spain) have been extracted and a pipeline featuring only reconciliation has been created. The measured average time per row was 12.927 ms. The same test was then repeated, enabling the caching level implemented at the reconciliation service level. The cache system improved performance achieving an average processing time of 2.558 ms per row (5 times faster over the baseline). Finally, we enabled the first cache layer, which is implemented locally on the level of the executable of the enrichment pipeline. The objective was to avoid the network latency whenever possible, which is substantial even in a local setup (via the loopback interface). The pipeline, in this case, ran $\sim$770 times faster than the baseline (0.0168 ms/row on average).

To analyze the behavior of the cache over time, a second experiment was designed, extending the first one as follows: a more complex pipeline (referred to as full-pipeline) was implemented. It reconciles city toponyms to GN, extends reconciled entities with the corresponding first-level administrative division from GN (i.e., regions). After that, it fetches weather information about regions, using the

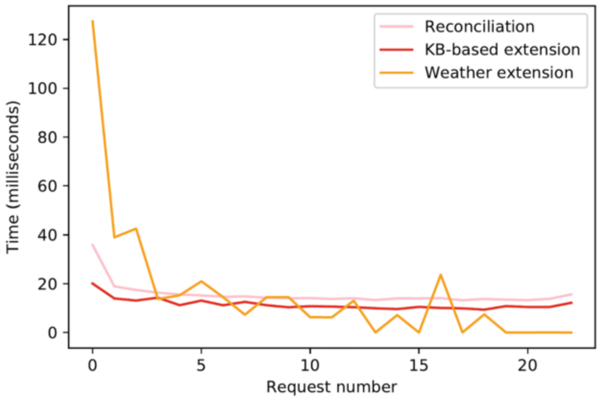


Fig. 6 Request execution time in milliseconds for the second experiment without duplicates

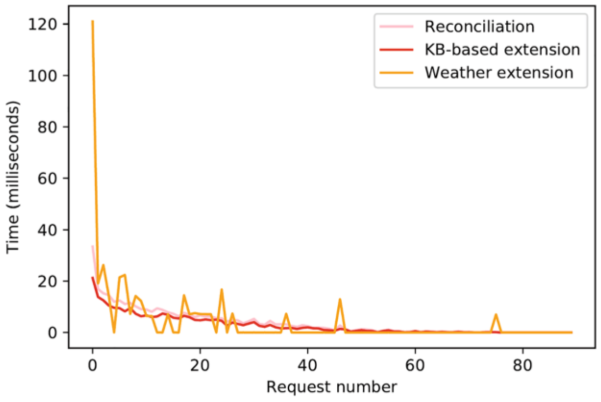


Fig. 7 Request execution time in milliseconds for the second experiment with four duplicates

reconciled administrative level and the date column (i.e., temperature for a specific date and the following one) generating a new dataset with 25 columns. This pipeline was used to enrich a dataset derived from the one used in the first experiment, filtering out duplicates in the reconciliation target column (i.e., each value occurs at most once), resulting in 2227 unique cities (and rows). The outcomes of this experiment, where the cache did not significantly improve the performance (as it was built but never used), are depicted in Fig. 6.[11] Afterwards, a synthetic dataset was built where each line from the previous one is replicated four times to exploit the local cache. As reported in Fig. 7, spikes are still visible due to cache building, but the cache reuse speeds up the process progressively (4× on average), considerably reducing the execution time (which tends to be purely cache access time).

[11] Initial spikes are due to the system startup (e.g., database connectors initialization).

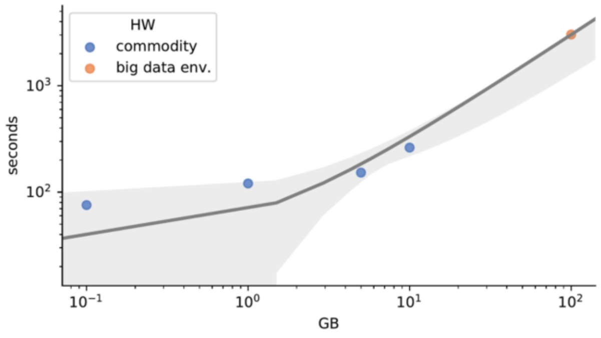


Fig. 8 Total execution time (in seconds) and linear regression curve, for different dataset sizes and two experimental setups

The final experiment was devoted to investigating the system scalability. Firstly, a single physical machine with a single instance of ASIA back-end was used. The full-pipeline was ran to enrich datasets of different sizes: 100MB, 1GB, 5GB, and 10GB. The dataset was split in 10 chunks of equal size and assigned to 10 agents. Performance results (in blue), reported in Fig. 8, measure the total pipeline completion time for different dataset sizes. The implementation achieves a linear trend, which highlights the scalability of the proposed solution. Finally, the enrichment of a ~100GB dataset (~500 million rows, 21 columns) was performed; the pipeline was run on the Big Data Environment deployed on a private cloud infrastructure featuring an 8-node cluster of heterogeneous hosts. Five of the nodes have 4-core CPUs and 15.4GB RAM and three nodes with 12-core CPUs, 64GB RAM, with six 3 TB HDDs holding a GlusterFS distributed file system (shared across the whole cluster). The enrichment agents were deployed on the three 12-core servers.

The transformation accessed a load-balanced (using round-robin load balancing) set of 10 replicas of ASIA back-end services deployed on the same stack.

The linear trend with $R^2 = 0.998$ (please notice Fig. 8 uses a base-10 log scale for the axes) is maintained also for the data point pertaining to the 100GB experiment, despite the different context in which the experiments have been carried out. This is mainly due to similar access and reconciliation times between the two experimental configurations.

5 Related Work

During the past decade, a number of tools devoted to tabular data transformation, reconciliation, and extension have been proposed. Probably the most popular is OpenRefine, an open-source solution that provides reconciliation services (with a focus on Wikidata) to semantic experts. It is a generic tool that cannot be easily customized to use third-party services. OpenRefine provides an interactive user interface with spreadsheet-style interaction in a similar way as Grafterizer does, hence they both encounter memory limitations to handle datasets. We refer the reader to [25] for a detailed discussion on data anomalies in tabular data together with an introduction to the table manipulation approach that was adopted in our work. OpenRefine was designed as a web application without support for batch execution of pipelines, hence the size of data that can be processed is constrained by the available memory. More recently, tools for extending OpenRefine with support to large data processing have been proposed, for example, OpenRefine-HD,[12] which extends OpenRefine to use Hadoop MapReduce jobs on HDFS. One of the few works on transformations of Big Data is discussed in [26], where the authors address how to exploit Apache Spark as a processing engine for iterative data preparation processes. However, all the above proposals require manual preparation of the executions, while in our approach we foresee an automatic deployment of the pipelines. An issue with all such tools is the lack of documentation that explains how the proposed solutions have been implemented, and how they can support scalability when working with distributed workloads, e.g., involving external services.

Semantic table annotation approaches have been proposed for reconciling values in tables; however, most of them cover only schema-level annotations. Approaches, such as [10, 27], are sophisticated and are targeted at Web tables, which are very small (a few hundred rows) and still require considerable computation time, making them inapplicable in Big Data environments. Karma [9] is a tool that provides an interface and a collection of algorithms to interpret tables, maps their schema to an ontology, and learns data transformations. However, Karma does not support external services for value-level reconciliation and data extension. The tool has been used in projects where these processing steps have been applied, but without explicit support by the tool itself [28]. Karma supports the execution of domain-specific scripts for data manipulations, i.e., to implement data cleaning tasks.

One of the commercial tools that are relevant to this work is Trifacta Wrangler,[13] which is a commercial suite of web applications for the exploration and preparation of raw datasets. The toolkit aims to provide data workers with specific smart tools to prepare datasets for different analysis types. Advanced techniques of machine learning, parallel processing, and human–machine interaction are also provided. The suite consists of three software solutions with increasingly advanced features. Large volumes of data can be handled by exploiting Cloud data warehouse deployments. KNIME[14] is a free software for analytics with a modular platform for building and executing workflows using predefined components called nodes. Knime core functionalities are used for standard data mining, analysis, and manipulation, and these features and functionalities can be extended through extensions from various groups and vendors. Big Data extensions allow the user to deploy workflows on Apache

[12] https://github.com/rmalla1/OpenRefine-HD

[13] https://www.trifacta.com

[14] https://www.knime.com

Spark and Hadoop clusters. Talend[15] is an Eclipse-based visual programming editor. Similar to KNIME, Talend uses predefined components (called nodes) to set up a pipeline, which can be compiled into executable Java code. Talend also provides an open-source data integration platform with Big Data extensions. None of these tools offer specific functionality for semantic enrichment. However, it is possible, using ad hoc methods (such as downloading data sources locally or using SPARQL queries), to approximate the functionality offered by ASIA. This result, however, can only be obtained by expert users and through specific code implementation. An extensive comparison of these tools with ASIA/DataGraft, where several factors are compared, including the number of correct, incorrect, and missing reconciliations, as well as the number of ambiguous toponyms correctly recognized, is available in [29], where the advantages of our solution in the reconciliation task are discussed in further detail.

6 Conclusions

In this work, we outlined and discussed an approach that addresses the efficient enrichment of massive datasets. The approach was developed as a result of the experience gained by closely working with business partners. We linked it to the practice by identifying the main challenges in the data science field, where actors need to integrate large datasets but often have limited programming expertise. Moreover, we proposed an open-source solution that features several enrichment services, which makes KBs accessible to non-expert users, supporting data enrichment both at the design and run time. Furthermore, repeatability is addressed by packaging the human expert actions within executable models, which can also be exploited to run the user-designed enrichment at a larger scale. The first implementation of our solution was deployed and tested in a real-world scenario. Preliminary experiments highlighted promising performance in terms of scalability; indeed, the prototype system was used to successfully execute a data flow to enrich data in the magnitude of hundreds of GBs continuously. In terms of future work, we plan to further improve the overall performance by addressing the limitations discussed in Sect. 3.2. We also plan to investigate more sophisticated solutions for entity linking in tabular data, which is particularly challenging in large tables [30]. To this end, we plan to develop approaches combining the feedback of possibly more than one users as proposed for analogous tasks [31].

Acknowledgments This work is partly funded by the EC H2020 projects EW-Shopp (Grant nr. 732590), euBusinessGraph (Grant nr. 732003), and TheyBuyForYou (Grant nr. 780247).

[15] https://www.talend.com

References

1. IDC. (2019). *Worldwide semiannual big data and analytics spending guide*. https://www.idc.com/getdoc.jsp?containerId=IDC_P33195
2. Zillner, S., Curry, E., Metzger, A., Auer, S., & Seidl, R. (Eds.). (2017). *European big data value strategic research & innovation agenda*.
3. Lohr, S. (2014). For big-data scientists, 'janitor work' is key hurdle to insights. *NY Times, 17*.
4. Furche, T., Gottlob, G., Libkin, L., Orsi, G., & Paton, N. W. (2016). Data wrangling for big data: Challenges and opportunities. In *EDBT* (pp. 473–478).
5. Čreslovnik, D., Košmerlj, A., & Ciavotta, M. (2018). Using historical and weather data for marketing and category management in ecommerce: The experience of EW-shopp. In *Proceedings of ECSA '18* (pp. 31:1–31:5). ACM.
6. Beneventano, D., & Vincini, M. (2019). Foreword to the special issue: "Semantics for big data integration". *Information, 10*, 68.
7. Koutsomitropoulos, D., Likothanassis, S., & Kalnis, P. (2019). Semantics in the deep: Semantic analytics for big data. *Data, 4*, 63.
8. Zhuge, H., & Sun, X. (2019). Semantics, knowledge, and grids at the age of big data and AI. *Concurrency Computation, 31*.
9. Knoblock, C. A., Szekely, P., Ambite, J. L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., & Mallick, P. (2012). Semi-automatically mapping structured sources into the semantic web. In *The semantic web: Research and applications* (pp. 375–390).
10. Ritze, D., Lehmberg, O., Bizer, C. (2015). Matching HTML tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS 2015*, Larnaca, Cyprus, July 13–15, 2015 (pp. 10:1–10:6).
11. Ermilov, I., & Ngomo, A. C. N. (2016). Taipan: Automatic property mapping for tabular data. In *Knowledge engineering and knowledge management* (pp. 163–179).
12. Kruit, B., Boncz, P., & Urbani, J. (2019). Extracting novel facts from tables for knowledge graph completion. In *The semantic web – ISWC 2019* (pp. 364–381). Springer.
13. Chabot, Y., Labbé, T., Liu, J., & Troncy, R. (2019). DAGOBAH: An end-to-end context-free tabular data semantic annotation system. In *Proceedings of SemTab@ISWC 2019. CEUR Workshop Proceedings* (Vol. 2553, pp. 41–48). CEUR-WS.org.
14. Nikolov, N., Ciavotta, M., & De Paoli, F. (2018). Data wrangling at scale: The experience of ew-shopp. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings* (pp. 32:1–32:4). ECSA '18, ACM.
15. Zillner, S., Bisset, D., Milano, M., Curry, E., Garcìa Robles, A., Hahn, T., Irgens, M., Lafrenz, R., Liepert, B., O'Sullivan, B., & Smeulders, A. (Eds.). (2020). *Strategic research, innovation and deployment agenda – AI, data and robotics partnership*. third release. Brussels. BDVA, EU-Robotics, ELLIS, EurAI and CLAIRE (September 2020).
16. Sukhobok, D., Nikolov, N., Pultier, A., Ye, X., Berre, A., Moynihan, R., Roberts, B., Elvesæter, B., Mahasivam, N., & Roman, D. (2016). Tabular data cleaning and linked data generation with grafterizer. In *ISWC* (pp. 134–139). Springer.
17. Cutrona, V., Ciavotta, M., Paoli, F. D., & Palmonari, M. (2019). ASIA: A tool for assisted semantic interpretation and annotation of tabular data. In *Proceedings of the ISWC 2019 Satellite Tracks. CEUR Workshop Proceedings* (Vol. 2456, pp. 209–212).
18. Roman, D., Nikolov, N., Putlier, A., Sukhobok, D., Elvesæter, B., Berre, A., Ye, X., Dimitrov, M., Simov, A., Zarev, M., Moynihan, R., Roberts, B., Berlocher, I., Kim, S., Lee, T., Smith, A., & Heath, T. (2018). Datagraft: One-stop-shop for open data management. *Semantic Web, 9*(4), 393–411.
19. Palmonari, M., Rula, A., Porrini, R., Maurino, A., Spahiu, B., & Ferme, V. (2015). ABSTAT: Linked data summaries with abstraction and statistics. In *ISWC* (pp. 128–132).
20. Stonebraker, M. (1986). The case for shared nothing. *IEEE Database Engineering Bulletin, 9*(1), 4–9.

21. Dessalk, Y.D., Nikolov, N., Matskin, M., Soylu, A., & Roman, D. (2020). Scalable execution of big data workflows using software containers. In *Proceedings of the 12th International Conference on Management of Digital EcoSystems* (pp. 76–83).
22. Wind, D. (2013). *Instant effective caching with ehcache*. Packt Publishing.
23. Fette, I., & Melnikov, A. (2011). The websocket protocol. Technical Report RFC 6455, IETF.
24. Sumaray, A., & Makki, S. K. (2012). A comparison of data serialization formats for optimal efficiency on a mobile platform. In *Proceedings of ICUIMC '12*.
25. Sukhobok, D., Nikolov, N., & Roman, D. (2017). Tabular data anomaly patterns. In *2017 International Conference on Big Data Innovations and Applications (Innovate-Data)* (pp. 25–34).
26. Wang, H., Li, M., Bu, Y., Li, J., Gao, H., & Zhang, J. (2015). Cleanix: a parallel big data cleaning system. *SIGMOD Record, 44*(4), 35–40.
27. Limaye, G., Sarawagi, S., & Chakrabarti, S. (2010). Annotating and searching web tables using entities, types and relationships. *PVLDB, 3*(1), 1338–1347.
28. Kejriwal, M., Szekely, P. A., & Knoblock, C. A. (2018). Investigative knowledge discovery for combating illicit activities. *IEEE Intelligent Systems, 33*(1), 53–63.
29. Sutton, L., Nikolov, N., Ciavotta, M., & Košmerlj, A. (2019). *D3.5 EW-Shopp components as a service: Final Release*. https://www.ew-shopp.eu/wp-content/uploads/2020/02/EW-Shopp_D3.5_Components-as-a-service_release_v1.1-SUBMITTED_Low.pdf
30. Cutrona, V., Bianchi, F., Jiménez-Ruiz, E., & Palmonari, M. (2020). Tough tables: Carefully evaluating entity linking for tabular data. In *ISWC*.
31. Cruz, I. F., Palmonari, M., Loprete, F., Stroe, C., & Taheri, A. (2016). Quality-based model for effective and robust multi-user pay-as-you-go ontology matching. *Semantic Web, 7*(4), 463–479.