Semantic Enrichment for Large-Scale Data Analytics*

Vincenzo Cutrona[0000−0001−7830−7596]

University of Milano - Bicocca, Milan, Italy
vincenzo.cutrona@unimib.it

Abstract. Most of all data science projects involve a time-costly data preparation process aimed at enriching the working dataset with additional information to improve the sturdiness of resulting trained models. How to ease the design of the enrichment process for data scientists is defying, as well as supporting the enrichment process at large scale. This document introduces and describes a research proposal for addressing such problem, which focuses on harnessing the semantics as the key factor, by providing users with semantics-aided tools to design transformations, along with a platform to execute pipelines at business scale.

Keywords: Big Data Processing · Data Integration · Data Enrichment · Data Extension · Linked Data

1 Problem statement

By 2020, the information will yield revenues in the order of 200 billion dollars [4], thanks to advancements in data science technology that have made it possible to analyze massive amounts of data and develop highly accurate and effective decision-making processes via analytical models. Such models require a huge amount of high-quality data to be robust enough, but despite the tremendous advancements achieved in processing, data sources are still filled with errors and inconsistencies. Moreover, many analyses are focused on studying variables that are not present in the main data source (e.g., weather-based analysis of digital marketing campaign performance), demanding the enrichment of the dataset with information from external sources (e.g., weather forecast web services).

The *data enrichment* is a specific data integration problem where a working data source, usually known to the data scientist, is enriched with additional information coming from external sources, usually less known to the data scientist. In this panorama, semantic techniques can support and simplify data enrichment. Nowadays the Linked Open Data (LOD) cloud provides several sources of information, like a Knowledge Base (KB), full of valuable information for data enrichment (high-quality and well-maintained data). Much of this information is

* I'd like to thank my PhD supervisor Prof. Matteo Palmonari for his comments and feedback. This work is funded by a scholarship granted by the Italian Ministry of Education, Universities and Research (MIUR).

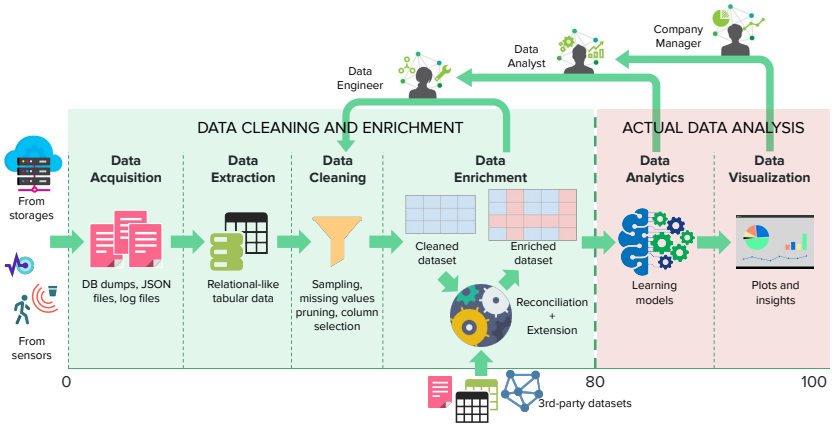


Fig. 1: Infographic representing the main stages of a typical data science project and the related stakeholders

open and accessible, but there are few attempts to use it to enrich tabular data. The *semantic data enrichment* harnesses the semantics to support the user in the enrichment task, relying on two main sub-tasks: the *reconciliation*, where values in the main source are linked to *entities* in external KBs, and the *extension*, where the identifier of linked entities are used to fetch data from external sources (which adopt the same space of identifiers) and extend the information in the main source.

As a matter of fact, a typical data science pipeline is mostly focused on the data preparation stage (Fig. 1), which cleans and enriches the data taking up to 80% of the time required by a project; only the remaining 20% is spent on data analysis [6]. This issue is now widely recognized and necessitates appropriate tools and methodologies because this imbalance foreshadows a problem that will worsen as the volume and variability of data increase [3].

Current approaches do not provide comprehensive solutions to the problem of data enrichment. Some solutions provide users with data-scientist-friendly tools to data preparation, supporting the data extension at a limited extent (i.e., requiring code scripts to link external sources), but without accounting the development/deployment of production-ready pipelines, hence they offer limited scalability. Otherwise, tools that support pipelines execution have been often designed for users with programming skills, but who are usually inexperienced in the particular domain to which the data pertain [11]. The main implication is the emergence of a two-pronged working environment consisting, on the one hand, of domain experts in charge of designing data transformations and, on the other hand, of engineers who deploy them into a production environment. This socio-occupational gap between interdependent groups pigeonholed in strictly separated roles can cause issues and delays in the development and maintenance of Big Data solutions and calls for specific solutions to be bridged. This research

project aims at providing a tool to support users in semantically enriching tabular data at large scale, both at schema- and instance-level (also referred as table annotation and instance matching, respectively, both covered by the Table interpretation field), in such a way to enable data extension.

2 Relevancy

This research proposal aims to deliver a scalable cloud-based solution to provide data scientists with the tools to (i) design transformation pipelines on tabular datasets that include data enrichment, harnessing the semantics to bring knowledge bases information (generic or specific) to non-expert people, and (ii) record and manage these pipelines in a repeatable form over large amounts of data, taking full advantage of the potential for scalability and distributed computing offered by the cloud. All people involved in data science projects could benefit from this solution, because at the same time it (i) eases the enrichment task, and (ii) make it repeatable in a scalable environment, reducing the overall data processing time. Besides, the gap between data engineers and analysts will be bridged: a single figure, the data scientist, will be able to design the transformation process and deploy it by herself, avoiding the continuous handover between engineers and analysts. At last, the proposed solution should ease the access to public knowledge bases also for non-expert people; in fact, so far knowledge bases can be mostly explored by people with a background in Semantic Web.

3 Related Work

Providing support for the design and execution of data transformation based on scale enrichment requires addressing different topics that have been investigated in the literature, leading to the provision of some tools dedicated to the transformation, reconciliation, and extension of tables.

Table transformation. We briefly discuss the table transformation literature because, even if it is not strictly related to this work, in many cases the table transformation is preliminary to the table interpretation tasks (almost all the table interpretation proposed methods perform a pre-processing phase to automatically clean the values in table). The table transformation deals with preliminary profiling and transformation of the tabular data and aims to identify and address possible data anomalies, eventually resulting in data that is shaped in a way that is easier to work with for further tasks (e.g., data reconciliation). A number of approaches and tools have been developed to deal with data anomalies, ranging from spreadsheet software (e.g., *Microsoft Excel*[1]), to programming languages and libraries for statistical data analysis (e.g., *Agate*[2] Python library for data analysis), to complex systems designed to be used for interactive data cleaning and transformation in ETL process (e.g., *Pentaho Data*

[1] `https://products.office.com/en/excel`
[2] `https://agate.readthedocs.org/en/1.3.1`

Integration[3] and *Trifacta Wrangler*[4]). These tools differ in the target audience and the capabilities they offer (e.g., coverage of data anomalies, mechanisms to address the anomalies). Recent research in this area focuses on intelligent and semi-automated mechanisms to detect and address data anomalies to simplify and automatize the data preparation process. A detailed discussion on the state of the art of this field is discussed in [12]. In the Big Data context, one of the few works on the transformation of massive datasets is discussed in [16], which addresses how to exploit Apache Spark for iterative data preparation processes.

Table interpretation. Most of research work has traditionally focused on schema-level alignment so [5, 7–10, 14, 15, 18]. Most of them create matches only for columns containing mentions of a real-world entity, while [18] and [8] focus also on literal columns. The schema-level alignment is useful and valuable to generate KBs and lift tabular data to RDF, but also to support the instance-level reconciliation. The latter is the key functionality in data science pipelines because it enables the extension of a source dataset. Interesting approaches to instance-level reconciliation have been proposed in the scientific literature. A few semantic table annotation approaches have been proposed for exploiting the semantics to reconcile values in tables, where most of them cover only schema-level annotations. Approaches like [5] and [10] are sophisticated and targeted for Web tables, which are very small (a few hundreds of rows), but require at the same time a lot of computations, making these approaches inapplicable in big data environments. A tool that provides an interface and algorithms to interpret tables, map their schema to an ontology, and learn data transformations is Karma [14]; however, Karma does not support external services for value-level reconciliation and data extension. Finally, we remark that the topic of data extension has not been addressed adequately in the scientific literature. Most of the approaches to fuse two tables have focused on conflict resolution strategies [1], i.e., how to deal with overlapping information. In data extension though, data to be added can be assumed to be new, and the conflict resolution problem may occur but is not the objective of the fusion operation. The most popular tool that supports instance-level reconciliation and extension is OpenRefine. It provides interactive user-interfaces with spreadsheet-style interaction embedded in a desktop application designed for people who are experts in semantics, and allows users to extend tables only with information contained in the same KB used for the reconciliation task (sameAs links cannot be used for entering a different KB), or by manually invoking third-party services with the string-content of a cell as parameter (i.e., it requires a short script to invoke an external service). Although the OpenRefine design phase is well supported, the tool comes without any support for batch execution of pipelines; thus it can only process data that can be entirely stored in memory, which is not suitable for the Big Data context. The community around OpenRefine proposed some tools for extending such tool with support to large data processing. Among them, the most remark-

[3] `http://community.pentaho.com/projects/data-integration`
[4] `https://www.trifacta.com/products/wrangler`

able is OpenRefine-HD[5], which extends OpenRefine to use Hadoop's MapReduce jobs on HDFS clusters. Unfortunately, documentation is missing for such a solution, and it is not stated how it can support scalability when more distributed datasets, exposed by external services, are involved.

To summarize, the application spotlighted in this work differs from the existing tools because (i) it aims at supporting all the above functionalities (table manipulation, reconciliation, and extension), bridging the gap left from existing tools that tend to focus on one or a subset of them; (ii) it foresees an automatic deployment of the pipelines created at design time, providing the user with an executable model that allows her to re-apply the same pipeline several times (i.e., applying the same pipeline to different datasets, which share the same schema but have different row values), while all the above proposals require manual preparation of the executions, and do not consider the repeatability of a task over different datasets.

4 Research Questions

The research project aims at addressing the following research questions:

RQ1: Are the current state of the art reconciliation approaches executable in a Big Data environment?
RQ2: Is it possible to insert the human in the enrichment process, in such a way to use her feedback to improve the system performance?
RQ3: Is it achievable to use the history of enrichment pipelines to give suggestions to the user about creating a new pipeline?

5 Hypotheses

The research questions stated in Section 4 will be addressed driven by the following hypotheses:

(RQ1) H1: State of the art reconciliation approaches can be executed in a Big Data environment, but they do not scale as the dataset size increases.
(RQ1) H2: Making state of the art reconciliation approaches scalable requires to rely on a smaller set of information, leading to worse performance.

(RQ2) H1: Users feedback can be learned with machine learning models (e.g., neural networks) and reused for subsequent executions to improve the performance.
(RQ2) H2: Enrichment steps can be executed in near-realtime on in-memory datasets to support the human-in-the-loop interactively.

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

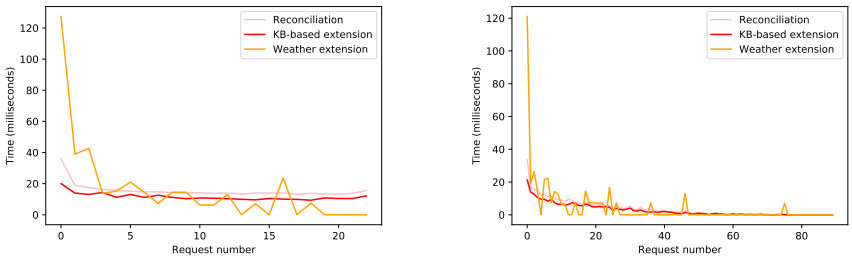
(RQ3) H1: Deep learning architectures can be used to learn how users design enrichment pipelines. Once the network is trained, it can be fine-tuned to support this task in different domains (similarly to transfer learning approaches).

6 Preliminary results

To obtain preliminary results that demonstrate the soundness of this project proposal, we started by extending Grafterizer, a tool that supports data transformation pipelines design and execution, with a new embedded application, ASIA (Assisted Semantic Interpretation and Annotation Tool), which supports users in semantically extending their data. ASIA has been designed as a set of micro-services, namely *reconciliation* and *extension services*, each one dedicated to reconcile/extend data to/with a specific KB. Those services are put behind a gateway, which allows users to use different KBs at a time to enrich their data. To test the flexibility and scalability of such architecture based on micro-services, we executed three experiments on real datasets; within all experiments, we designed an enrichment pipeline that extends the dataset by querying a weather KB, by reconciling and extending city toponyms to a geospatial service (based on GeoNames (GN)). Both GN and the weather KB have been exposed as distributed services, thus ASIA must perform a series of HTTP requests for reconciling and extending the dataset.

First, we simulated the scenario where the data scientist executes the enrichment pipeline on a commodity machine (small-scale). The main objective was to assess how much the performance is affected by HTTP requests in a distributed environment, and how much it boosts the system performance by adding different levels of cache. We started by testing the reconciliation performances with no caching strategy whatsoever: 200 thousand rows from a real company dataset featuring 2227 different toponyms (from Germany and Spain) have been extracted and a pipeline featuring only reconciliation executed. The measured average execution time per row was 12.927ms. The same test has been then repeated enabling a cache implemented at the reconciliation service level. This cache system improved the performances achieving on average 2.558ms per row (5x times faster with respect to the previous baseline). At last, a second cache layer has been enabled, which is implemented locally on ASIA. The objective is to avoid the network latency, 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 (0.0168ms/row on average).

In order to analyze the behavior of the cache over time, a second experiment has been designed extending the first one as following: a more complex pipeline is considered, which reconciles city toponyms to GN, extends reconciled entities with their first administrative level (i.e., regions), and fetches weather information about regions (i.e., temperature for a specific date and the day after) generating a new dataset with 25 columns. This pipeline has been employed first to enrich a dataset derived from one of the first experiments filtering out the



(a) Without duplicates. (b) With 4 duplicates.

Fig. 2: Request execution time in milliseconds for the second experiment.

duplicates in the reconciliation target column (i.e., each value occurs at most once); thus, 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. 2a[6]. Afterward, a synthetic dataset was built where each line from the previous dataset is repeated four times, allowing to exploit the local cache. As reported in Fig.2b, spikes are still visible due to cache building, but the cache reuse speeds up the process progressively (4x on average), reducing the execution time (which tends to be purely cache access time) considerably.

The final experiment was devoted to investigate the system scalability. First, a commodity machine is used (this experiment like the previous ones have been performed on a multi-tenant machine with 4 CPUs Intel Xeon Silver 4114 - 2.20GHz, and 125GB RAM) and ASIA deployed singularly. The same pipeline was used to enrich datasets of different size: 100MB, 1GB, 5GB, and 10GB, divided into 10 chunks of equal size and assigned to 10 *agents* (i.e., components that execute the pipeline). Performance results (in blue), reported in Fig. 3 as dataset size/total completion time, show 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 3TB HDDs holding a GlusterFS distributed file system (shared across the whole cluster). The enrichment agents were deployed on the three servers. The transformation accessed a load-balanced (using round-robin load balancing) set of 10 ASIA services deployed on the same stack. The linear trend with R^2=0.998 is maintained also when the 100GB experiment is considered (the orange point in Fig. 3), 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 configurations used.

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

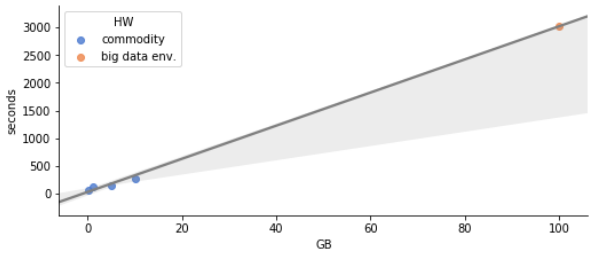


Fig. 3: Total execution time (in seconds) and the linear regression line, for different dataset sizes and two experimental hardware (HW) setups.

7 Approach

The approach proposed in this research project to facilitate data processing and enrichment at scale is inspired by a *small-scale design/full-scale execution* principle, harnessing the semantics to support reconciliation operations that are required for data enrichment. The high-level description of this principle can be sketched as in Fig. 4. The overall design decision is to separate the transformation process in two phases: the *design phase*, where the user defines the transformation pipeline working on a sample of the original dataset, and produce an *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 the enriched dataset to feed the analytical activities. Both phases rely on distributed datasets and services to support reconciliation and extension activities. In the data science context, this separation brings another advantage in terms of privacy and security: pipelines can be designed on a small sample, keeping private the entire company dataset. At last, since the design phase exploits only a sample, the limited quantity of the data to be transferred and manipulated simplify the infrastructure required to execute the

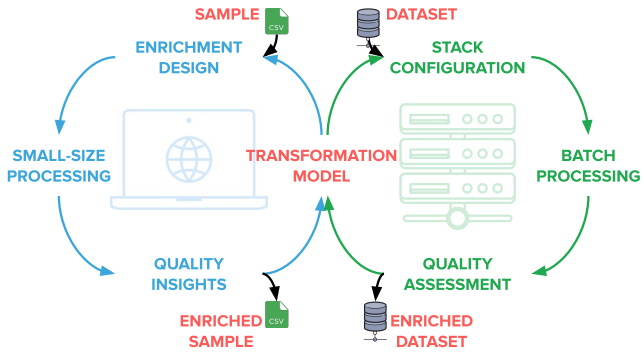


Fig. 4: The Design/Processing approach at a glance

enrichment tools (i.e., they can run in browsers on regular laptops). Instead, the execution phase can be enacted on-premise, thus exploiting the actual corporate infrastructure and tools, without moving the original dataset.

The data enrichment solution has been conceived to obtain large volumes of value-adding data to train analytical models in the easiest way (i.e., without requiring comprehensive programming skills). Most of all approaches available in state of the art are fully-automated approaches that are not suitable for this purpose, even if they represent the easiest way to transform the data, since they would entirely remove the human control over the process and results. Consequently, the above principle allows to give the human control over the quality of the results (e.g., she can fix errors introduced by the reconciliation) during the design phase, and then her choices are packaged within an enrichment pipeline runnable in batch over a different (possibly massive) dataset.

7.1 Design phase

In the design phase the user is involved in the design of the pipeline, by interactively performing three steps: (i) the *enrichment pipeline design*, where the transformation pipeline is built by means of a graphical interface that facilitates and automates interactions with reconciliation and extension services, (ii) the pipeline execution (*small-size processing*), where each step of the enrichment pipeline is performed over the small-sized data, and the (iii) *quality insights* evaluation, where the user is provided with a few statistics that give a general understanding of the overall quality of the result (e.g., 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); thus, when the user is required to enrich considerable volumes of data (or at least larger than can be interactively managed), this approach envisages that the user carries out the design phase using a representative sample of the original dataset. It is essential to point out that the end-user might not necessarily proceed to the processing stage in the case when she needs to enrich tables with a few thousand rows, which can be achieved in the design phase.

Finally, the user-defined enrichment pipeline is turned in an executable transformation model (e.g., packaged in a Java archive (JAR)), which can be downloaded from the user interface and reused as the main step within the processing phase.

7.2 Processing phase

The goal of this phase is to execute the pipeline designed and tested during the previous stage on a smaller (loadable in memory) dataset on a large (Big Data) dataset through parallelization.

As in the previous phase, three steps can be identified: (i) the data flow definition (*Stack configuration*), which includes the enrichment pipeline, (ii) the

batch execution (possibly in parallel) of the pipeline, and (iii) the *quality assessment* of the resulting dataset. If the result does not achieve an acceptable quality level, the user could go back and design a new pipeline.

As data flows we refer to an extension of the enrichment pipeline with pre- and post-processing actions, such as decompression of the input table from user archives, creation of chunks for parallelizing the process, and so on.

It is important to underline that the services implementing the reconciliation and extension functionalities must be available to be invoked (as a service) as the KBs essential to their operations can be massive, making impracticable to encapsulate them within the executable transformation model. This architectural choice, which derives from precise requirements such as modularity and flexibility, constitutes the main limitation preventing the scalability of the entire enrichment process.

8 Evaluation plan

The outcome of this research project will be a tool for supporting data scientists in enriching their datasets with third-party sources, at large scale. Many features must be evaluated to understand the success of the proposed approach:

- **reconciliation performance**: the reconciliation services should be compared with state of the art approaches in terms of precision and recall; a lower performance is expected, due to the high complexity of state of the art approaches that is not sustainable at large scale. The best tradeoff between performance and execution time should be found (**RQ1 H1**).
- **system scalability**: the preliminary results discussed in Section 6 show that the current system can scale, but this condition must be verified again when more sophisticated data reconciliation service will be adopted (**RQ1 H2**).
- **human-feedback learning capability**: quality insights given to the user at design time should be compared with the quality assessment results made after the large-scale execution phase, in order to assess the learning capability of the adopted machine learning models (**RQ2 H1**).
- **execution time**: the transformation and enrichment pipeline steps must be performed in near-realtime in the small-scale case, in such a way to guarantee interactivity in the design phase (**RQ2 H2**).
- **transfer learning**: transformation models can be used to train a network and learn how users design enrichment pipeline. Given a set of enrichment pipelines performed over a dataset that pertain to a specific domain (e.g., marketing), new pipelines for enriching datasets from a different context (e.g., e-commerce) can be automatically created using such network. The suggested pipeline must be compared with a human-designed pipeline designed for the same dataset (**RQ3 H1**).

9 Reflections

The objective of this research project is to provide a scalable system able to support users with semantic enrichment. Unlike other state of the art approaches,

which start by designing the most effective algorithm, and consider scalability issues only later, we act in the opposite way; we proceeded bottom-up by designing a scalable architecture, based on micro-services, and then we increment the usage of semantic techniques until the scalability property of our system holds. We believe that the scalability property is the most important if we want to push this system towards a different user base, i.e., data scientists, who often operate on very large datasets. For this reason, in this section, the lessons learned and the current limitations are discussed, along with some aspects that could be improved to bring a not marginal improvement to the performance of the entire process.

Data Locality. In the initial release of the Big Data Environment, the data locality principle is limited to the life-cycle management of data of the KB employed for enrichment that are brought into the environment; thus, reconciliation algorithms have direct access to the KB. Similarly, the working data are stored in a distributed file system and accessible through the network. This architectural choice uniforms access times to data, but also raises the average read/write times by twice the network latency. There is the chance to improve the performance by distributing the chunks among the machines that execute the agents' containers.

Distributed Caching. The current caching system has the main issue of being local, i.e., attached to each ASIA replicated deployment. Since the load balancer runs a round-robin dispatching policy, it occurs that identical requests are assigned to different replicas of ASIA, causing preventable cache misses. A better solution entails 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 (for example Ehcache [17]) would reduce the number of misses, guaranteeing a rapid synchronization of the content of the local caches.

Efficient API interaction. At present, for both the design and processing phases, reconciliation and extension are invoked for every single row of the working table. As a consequence, for each line the agent running the pipeline waits a time equal to the Round-trip Delay Time (RTD) for each line, forcing the system to wait time equal to twice the network latency for each line. Grouping the invocations to the service would improve the performance considerably. The processing times of the input dataset could be further improved if light network protocols (such as Websocket [2]) were used together with APIs that better exploit message serialization (such as Google Protobuf [13]).

References

1. Bui-Nguyen, Q., Wang, Q., Shao, J., Vatsalan, D.: Repairing of record linkage: Turning errors into insight. In: EDBT. pp. 638–641 (2019). https://doi.org/10.5441/002/edbt.2019.75
2. Fette, I., Melnikov, A.: The websocket protocol. RFC **6455**, 1–71 (2011). https://doi.org/10.17487/RFC6455

3. Furche, T., Gottlob, G., Libkin, L., Orsi, G., Paton, N.W.: Data wrangling for big data: Challenges and opportunities. In: EDBT. pp. 473–478 (2016). https://doi.org/10.5441/002/edbt.2016.44
4. IDC: Worldwide semiannual big data and analytics spending guide (2017), `https://www.idc.com/getdoc.jsp?containerId=prUS42371417`
5. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. PVLDB **3**(1), 1338–1347 (2010). https://doi.org/10.14778/1920841.1921005
6. Lohr, S.: For big-data scientists, 'janitor work' is key hurdle to insights. New York Times **17**, B4 (2014)
7. Mulwad, V., Finin, T., Joshi, A.: Semantic message passing for generating linked data from tables. In: ISWC. pp. 363–378 (2013). https://doi.org/10.1007/978-3-642-41335-3_23
8. Neumaier, S., Umbrich, J., Parreira, J.X., Polleres, A.: Multi-level semantic labelling of numerical values. In: ISWC. pp. 428–445 (2016). https://doi.org/10.1007/978-3-319-46523-4_26
9. Pham, M., Alse, S., Knoblock, C., Szekely, P.: Semantic labeling: A domain-independent approach. In: ISWC (2016). https://doi.org/10.1007/978-3-319-46523-4_27
10. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML tables to dbpedia. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS. pp. 10:1–10:6 (2015). https://doi.org/10.1145/2797115.2797118
11. Sukhobok, D., Nikolov, N., Pultier, A., Ye, X., Berre, A., Moynihan, R., Roberts, B., Elvesæter, B., Mahasivam, N., Roman, D.: Tabular data cleaning and linked data generation with grafterizer. In: ESWC. pp. 134–139. Springer (2016)
12. Sukhobok, D., Nikolov, N., Roman, D.: Tabular data anomaly patterns. In: IEEE Big Data Innovations and Applications. pp. 25–34 (2017). https://doi.org/10.1109/Innovate-Data.2017.10
13. Sumaray, A., Makki, S.K.: A comparison of data serialization formats for optimal efficiency on a mobile platform. In: ACM IMCOM. pp. 48:1–48:6 (2012). https://doi.org/10.1145/2184751.2184810
14. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. Journal of Web Semantics **37-38**, 152 – 169 (2016). https://doi.org/10.1016/j.websem.2015.12.003
15. Venetis, P., Halevy, A.Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. PVLDB **4**(9), 528–538 (2011). https://doi.org/10.14778/2002938.2002939
16. Wang, H., Li, M., Bu, Y., Li, J., Gao, H., Zhang, J.: Cleanix: a parallel big data cleaning system. SIGMOD Record **44**(4), 35–40 (2015). https://doi.org/10.1145/2935694.2935702
17. Wind, D.: Instant effective caching with ehcache. Packt Publishing Ltd (2013)
18. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer$^+$. Semantic Web **8**(6), 921–957 (2017). https://doi.org/10.3233/SW-160242