On the visualization of semantic-based mappings

Nicolò Oreste Pinciroli Vago[1,3], Mario Sacaj[1,4], Mersedeh Sadeghi[5], Safia Kalwar[1], Andreas Vogelsang[5], and Matteo Rossi[2]

[1] Dipartimento di Elettronica, Informazione e Bioingegneria – Politecnico di Milano
[2] Dipartimento di Meccanica – Politecnico di Milano, Milan, Italy, e-mail: {firstname.lastname}@polimi.it
[3] NTNU – Norwegian University of Science and Technology, Ålesund, Norway
[4] TUB – Technische Universität Berlin, Berlin, Germany
[5] Software and Systems Engineering, University of Cologne, Cologne, Germany, e-mail: {lastname}@cs.uni-koeln.de

Abstract. The popularity of the semantic web in many domains, such as transportation, has led to an ever-increasing development of standards, vocabularies, and ontologies, which generates problems of heterogeneity and lack of interoperability. To address this issue, a large body of research focused on providing various mapping tools and techniques to translate data from one standard to another to foster smooth communication among them. While valuable advancements in mapping techniques have been achieved so far, the explainability and usability of such tools have been overlooked. Since explainability of software is being recognized as a crucial non-functional requirement for complex systems, the development of self-explaining and user-friendly graphical interfaces is becoming a pressing need. In this paper we present S2SMaT, our contribution to the problem of visualization of mappings. The tool helps users easily navigate the structure of standards, understand the suggested mappings between their terms, and in general more easily interact with the system.

Keywords: Visualization · Coordinated views · XML to Ontology mapping · Automated mappings · Semantic Mappings · Visual Explanation

1 Introduction

As the benefits of the use of semantic web technologies in interoperation, knowledge management, and data retrieval become more evident, their popularity and application are growing in many domains. In particular, ontologies can significantly improve the interoperability of data-intensive and collaborative applications that exchange, share, and use a wide range of heterogeneous data. In this direction, the mobility and transportation domains have shown great interest in using ontologies as a tool in different application areas, from data simulation

and analysis to integration and interoperability of heterogeneous transportation data [1,2,3]. As a result, we are witnessing the emergence of an increasing number of co-existing ontologies, vocabularies, and data models, which are, in many cases, organization- and application-specific [4,5].

Therefore, to foster the interoperability of large, distributed systems that rely and operate on such a diverse set of ontologies, two ingredients are key. First, parties and systems using different ontologies should be able to interpret, understand, and smoothly interoperate with other parties' data models. Second, the gap between ontologies and non-ontological data sources and standards (e.g., well-known data models such as GTFS,[6] which has also an XML-based format) must be bridged to boost the usability of ontologies and semantic technologies in practice. To address these concerns in different domains, many mapping tools have been developed focusing on finding similarities and shared concepts between ontologies and standards represented in other formats (see Section 2).

Nevertheless, only little attention has been paid to making mapping suggestions more explainable. Indeed, interpretability and explainability in intelligent systems are growing concerns, particularly in machine learning-based applications [6,7]. The need for explaining the system behaviour increases when it involves some decision-making process or offers some suggestions and recommendations [8,9]. Many studies showed that explainability increases the trust of users in the system and helps them follow such decisions and suggestions more confidently, which leads to higher user satisfaction and engagement with the system [7,10]. In this regard, visualization and interactive user interfaces are known as a popular and effective approach toward making a system explainable [11,12].

The work presented in this paper focuses on the explainability of heterogeneous data mappings through visualization. More precisely, we present an extension of the ongoing research on the development of a mapping tool that is part of the Shift2Rail Interoperability Framework (IF) [13]. The mapping tool uses machine learning and linguistic matching techniques to find semantically similar concepts of any two given standards. In previous works [14,15] we introduced in detail the mechanisms—and underlying algorithms—for creating mapping suggestions. In this paper, we build on these mechanisms to create a tool, called S2SMaT, that combines (i) an interactive user interface that allows users to suitably visualize—and possibly modify—suggested mappings with (ii) mechanisms to automatically generate annotations capturing the identified mappings. S2SMaT is a web-based tool that supports coordinated views of graphs of concepts. The coordinated views approach has been beneficial in several diverse cases, such as simulated games [16], geospatial data visualization [17] and user behaviour analysis [18]. The aim of the S2SMaT tool is to increase the usability and transparency of the mapping tool and to make it more explainable.

The rest of this paper briefly overviews related works in Section 2, then describes the S2SMaT tool in Section 3 along with its protype implementation Section 4, and concludes in Section 5.

[6] https://gtfs.org/(As of July 2021).

2 Related Works

Mappings between ontologies and XML data representations are gaining more and more attention mainly due to the upsurge of data heterogeneity, as well as the development of domain-specific data representations, vocabularies, and ontologies [19,20]. For example, [21] offers a mapping approach between XML Schema and OWL ontology elements. [22] presents some RDF rules to enrich and populate existing ontologies given XML data. Yin et al. [23] combine context and word similarity algorithms to build an efficient ontology mapping framework. DTD2OWL [24] uses structural rules and adds semantics to XML documents to create an automated transformation of XML to OWL ontology [25]. In the works mentioned above and many other similar contributions in data mappings [26,**?**,27,28], a common drawback is a lack of visualization support for the mapping process or results. While their main contribution to enhancing the effectiveness and performance of the overall transformation process is valuable, they left the traceability and explainability requirements behind.

Few contributions, such as [21] and [29] developed GUIs for their proposed mapping process. However, their presentation of the XML and ontology files follows a simple hierarchy format, whereas in our case the files are visualized as collapsible and fully explorable trees/graphs. Furthermore, compared to their works, our GUI is more interactive and provides a more extended set of features such as searching a term, leading to an automatic zoom into the actual location of the word in the graph/tree. Finally, to improve the tool usability the overall design of our GUIs has followed the Gestalt principles [30].

3 S2SMaT

Figure 1 depicts the high-level architecture of the S2SMaT tool, which is composed of the *Computation Module* and *Interaction Module*, as well as the *Input Data Parsing* and *Output Generation* modules. The main focus of this paper is on the *Interaction Module* and on the *Graph Computation* sub-module of the *Computation Module*. In a nutshell, the tool takes as input two files, an XSD file and an ontology, computes a set of suggested mappings between concepts defined in the two files, presents the suggestions to the user in a user-friendly manner and, once the user confirms the mappings, generates suitable annotations that are compatible with the conversion approach defined in [31]. In the following, we briefly overview each component and the overall workflow of the system.

Data Parsing. Once the input XSD and ontology files are uploaded by the user, the *Input Data Parsing* phase starts. The system first checks the syntactic validity of the inputs, then the files are parsed, pre-processed, and cleaned to make them suitable inputs for the *suggestion computation*, graph visualization and annotation generation steps. Furthermore, the tool proceeds with a simple structural decomposition of the XSD and ontology files, which creates a representation that binds each term defined in the files to its respective syntactical type—i.e., Complex Type, Element and Attribute in XSD, and Class and

Property in the ontology. This so-called *binding representation* is later used to validate the mapping suggestions so only structurally equivalent terms can be mapped to each other. In other words, a term that is positioned as Complex Type or as an Element/Attribute in the XSD file should be respectively mapped to a Class or Property in the ontology.

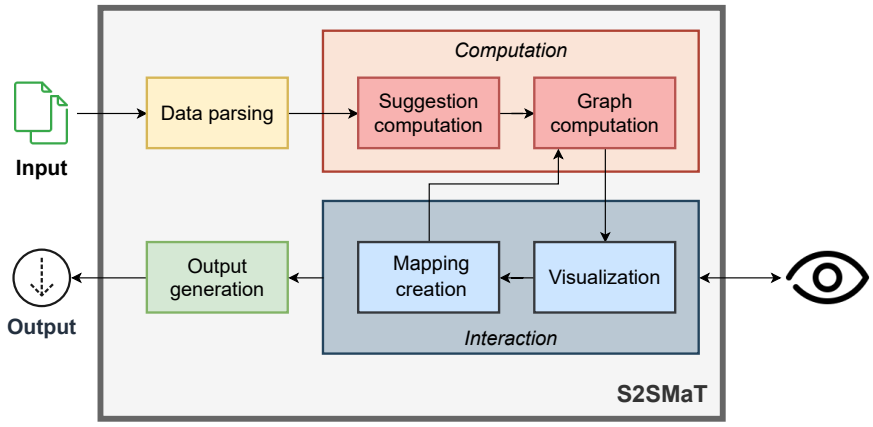

Fig. 1. S2SMaT architecture

Suggestion computation. As depicted in Figure 1, the *Computation Module* has two sub-modules, namely *Suggestion Computation* and *Graph Computation*. The former encompasses the Mapping Tool, which is one of the main utilities of the Interoperability Framework developed within the SPRINT project.[7] SPRINT aimed at fostering the seamless, semantic-based and secure interoperability among distributed organizations in the transportation domain [32], by offering a set of innovative services and tools such as ontology management, data converters, personalized travel companion, etc. [33,34,35,36].

The *Suggestion computation* module in S2SMaT currently incorporates the first version of the Mapping Tool,[8] which generates a one-to-one mapping between the concepts in an XSD specification and those in an ontology. In a nutshell, the module uses a Word2vec-trained model (*w2v*) [37] to compute the similarities of terms of the given standards. The *w2v* transforms each word appearing in a corpus to a 300-dimensional feature vector. Then, these vectors can be used to establish meaningful associations among words. More precisely, semantically similar concepts are identified based on the relative distances of the corresponding vectors in the space. The *Suggestion computation* component takes the two input files and, for each term in each file, extracts the topmost

[7] sprint-transport.eu (As of July 2021).

[8] See D4.3 - A lightweight solution to automate the generation of ontologies, mappings and annotations (F-REL) for further details (As of July 2021).

similar terms according to the pre-trained model (which is freely available in the literature, and was created based on the Google News dataset [38]). The module computes the similarity between pairs of terms (one for each input file) based on the number of similar words shared between them and on their *w2v* vector values. Finally, the pairs with a similarity value above a certain threshold are considered as matched pairs. The output of this component is a list of suggested mappings (i.e., pairs) of terms, one from the XSD file, and one from the ontology.

S2SMaT then inspects the list of suggestions against the types of each term given the *binding representation* and filters out structurally incompatible mappings (e.g., if an XSD Complex Type has been suggested for a Property in the ontology). The mappings are sent to the *Graph computation* and *Interaction* modules, which offer an interactive Graphical User Interface (GUI) to visualize and manipulate, in a user-friendly way, the XSD, the ontology and the suggested mappings between terms.

Interactive Visualization of Mappings. To make mapping suggestions more tangible and explainable for users, S2SMaT offers rich visualizations of the relevant aspects. Firstly, it provides a tree and a graph representation of the XSD and ontology files, respectively, making them easier to read, explore, and navigate. More precisely, the XSD file is displayed as a collapsible tree that encodes types of terms using different colours. The tree is fully interactive so users can expand/collapse the children nodes, zoom in/out of the nodes and view more details about each term by clicking on it. Furthermore, the GUI provides a searching capability where users can look for particular words to locate them in the tree. Similarly, the ontology is presented as a (possibly disconnected) graph with a search functionality and standard visualization options related to the distribution of nodes in the graph and its collapsing degree.

Finally, S2SMaT provides users with a GUI for viewing, manually inspecting, and modifying the list of mappings between the terms of two input data representations. In particular, users can select terms belonging to any of the two input files from a list of terms to view its suggested mapping in the other data representation. Additionally, by clicking on each term in the tree and graph visualizations, users can trace which term, if any, is currently mapped to the selected term and possibly entirely remove such mapping. The system also enables users to manually add new mappings, if necessary. In the end, a confirmed set of paired terms is sent to the *Output Generation* module.

Output generation. When users confirm a list of mappings, the system starts the output generation phase, which includes *annotations* creation and export.

S2SMaT offers an automated mechanism for the creation of Java-based annotations that materialize the suggested mappings. Annotations provide metadata about the Java elements (e.g., classes and methods) in a structured manner. Java annotations pragmatically represent suggested mappings between concepts in the two data representations and make them amenable to automated processing by external tools (in particular converters based on the mechanisms defined in [31]). S2SMaT first translates the elements of the XSD representation

in equivalent Java constructs. In particular, XSD's Complex Types, Elements and Attributes are transformed to Java classes, attributes, and setter/getter methods. These Java constructs are then annotated by the respective mapped term in the ontology using special-purpose annotations. For example, if the *Suggestion computation* component determines that the term `GeoCoordinate` in the IT2Rail ontology[9] should be mapped to the `GeoPoint` concept in the FSM standard,[10] where `GeoPoint` is a Complex Type in FSM, then the *Annotation Creation* component creates a Java class named `GeoPoint` and annotates it by the `@RdfsClass("IT2Rail:GeoCoordinate")` annotation. The annotated Java code is written to disk at the end, and the zipped file is generated as the final output.

4 Prototype implementation

The prototype implementation of the S2SMaT tool follows a client-server architecture using Java and Python for the server-side and JavaScript for the client-side. Figure 2 shows the screen viewed by the user after input files have been selected and a list of mappings has been generated by the *Suggestion Computation* module. The view is composed of three window-like boxes, in which the tree visualization of the XSD file, graph visualization of the ontology, and the respective mappings are shown. For the visualization of the standards S2SMaT integrates and extends some external tools and libraries. In particular, the windows management and style are based on the INTEGRA framework,[11] and the graph representation of the ontology is built on WebVOWL [39]. However, since the graphical renderer of the latter only accepts VOWL-formatted ontologies, we integrated the `owl2vowl`[12] tool to execute such task.

As mentioned above, the development of S2SMaT's interface is inspired by the Gestalt principles [30]. More precisely, to provide users with a predictable and self-explaining interface, controls with similar functions are grouped together and the icons have been kept consistent across the application following the *de-facto* standards in web development. In addition, we avoided designing any complex sequences of actions: the tool allows the user to keep the entire interface under control without memorizing past steps and without the need to navigate multi-level menus. Moreover, the provided visualization allows the self-organization of the graphs optimally, enabling users to navigate complex structures easily. Finally, the GUI is fully interactive and provides various facilities for users to explore the data and the suggested mappings and modify them.

Figure 3 shows an exemplar scene of one of our test cases, where a user has selected four mappings in the middle window (association window), so the respective terms in XSD (here the FSM standard) and ontology (here IT2Rail) files are visualized on the other windows within the enclosed tree and graph. The association window and tree/graph windows allow navigating the graph and

[9] http://it2rail.eu/ (As of July 2021).
[10] https://tsga.eu/fsm/ (As of July 2021).
[11] https://github.com/nicolopinci/INTEGRA (As of July 2021).
[12] https://github.com/VisualDataWeb/OWL2VOWL (As of July 2021).

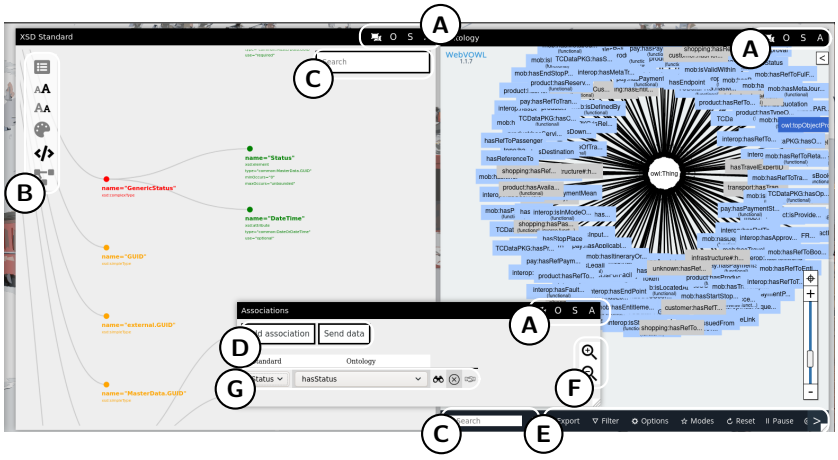

Fig. 2. Overview of the S2SMaT's interface – The tool interface is formed by several areas: (A) is used to choose which window must be maximized and to bring the current window on top; (B) contains a set of controls related to the standard tree visualization; (C) is used to perform a search in one of the graphs; (D) contains the controls related to the creation of annotations; (E) contains a set of controls related to the ontology graph visualization; (F) contains the zoom in and zoom out controls for the association window; (G) contains the list of created associations.

the tree by using mouse-based and keyboard-based interactions. In particular, the user can look for a specific term of interest using the search box in both windows so that the relevant words will be highlighted, and a zoom-in will adapt automatically. Alternatively, the user can navigate through the tree and graph simply by dragging and zooming to the desired position. Finally, while a mapped pair is selected, the user can modify either side of the mapping by relating one term to a new word from the other data representation (if it is structurally compatible), or entirely remove the association.

To benefit from the advantages of a modular architecture, the windows mentioned above have been developed as separate HTML pages, and a central client-side script manages the interactions among them. Furthermore, where useful, the components have been developed as independent and self-contained modules, which are then integrated into S2SMaT. More specifically, in addition to the *Suggestion computation* component, which is an external tool, we have developed two more stand-alone modules, namely the *OntologyConverter*[13] and *Annotator Tool*.[14] The former is a simple Java wrapper employing the OWL API library[15] designed to make the system compatible with Turtle-encoded ontologies, which is among the most popular ontology formats. The latter is a Java application and handles the annotation generation of S2SMaT. It exploits

[13] https://github.com/mskx4/OntologyConverter (As of July 2021).
[14] https://github.com/mskx4/AnnotatorTool (As of July 2021).
[15] http://owlcs.github.io/owlapi/ (As of July 2021).

Fig. 3. A view of S2SMaT representing the result of a test case to generate a mapping between the FSM standard (XSD) and IT2Rail Ontology. The window in the top-left corner contains the visualization of the FSM standard, the window in the top-right corner contains the visualization of the Ontology, and the window on the bottom contains the list of created associations.

`JAXB`[16] and `Jakarta XML Binding`[17] APIs to generate a Java source code given a well-formatted XSD file and ultimately annotates such Java structs based on the suggested mappings as explained in Section 3.

5 Conclusions

This paper presents S2SMaT, a tool for the automatic creation of mappings among terms and concepts in a pair of standards. It articulates the results as Java-based annotations, which facilitates further automated processing and utilization of such mappings. The main contribution of the paper is on the visualization of various aspects of the mapping process to increase the transparency and explainability of the overall procedure for the end-users. It offers a coordinated view of the graph of concepts and a set of rich and self-adaptive GUIs to visualize the suggested mappings and allow users to inspect and possibly modify the suggestions interactively. A prototype of the tool and preliminary experiments with well-known standards and ontologies in the transportation domain witness interesting results and motivate further works.

[16] https://github.com/eclipse-ee4j/jaxb-ri/ (As of July 2021).
[17] https://github.com/eclipse-ee4j/jaxb-api (As of July 2021).

References

1. Stijn Verstichel et al. Efficient data integration in the railway domain through an ontology-based methodology. *Transportation Research*, 19(4), 2011.
2. Thiago Sobral, Teresa Galvão, and Jose Borges. An ontology-based approach to knowledge-assisted integration and visualization of urban mobility data. *Expert Systems with Applications*, 150, 2020.
3. Filippo Benvenuti, Claudia Diamantini, Domenico Potena, and Emanuele Storti. An ontology-based framework to support performance monitoring in public transport systems. *Transportation Research, Emerging Technologies*, 81, 2017.
4. Megan Katsumi and Mark Fox. Ontologies for transportation research: A survey. *Transportation Research Part C: Emerging Technologies*, 89, 2018.
5. Ali Yazdizadeh and Bilal Farooq. Smart mobility ontology: Current trends and future directions. *arXiv preprint arXiv:2012.08622*, 2020.
6. Alejandro Barredo Arrieta et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Inf. Fusion*, 2020.
7. Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proc. of the SIGCHI Conf. on human factors in computing systems*, 2009.
8. Ingrid Nunes and Dietmar Jannach. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, 27(3), 2017.
9. Mersedeh Sadeghi, Verena Klös, and Andreas Vogelsang. Cases for explainable software systems: Characteristics and examples. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021.
10. Pearl Pu and Li Chen. Trust building with explanation interfaces. In *Proc. of the 11th Int. Conf. on Intelligent user interfaces*, 2006.
11. Hao-Fei Cheng, , et al. Explaining decision-making algorithms through ui: Strategies to help non-expert stakeholders. In *Proc, chi Conf. on human factors in computing systems*, 2019.
12. Sule Anjomshoae et al. Explainable agents and robots: Results from a systematic literature review. In *18th Int. Conf. on Autonomous Agents and Multiagent Systems*. Int. Foundation for Autonomous Agents and Multiagent Systems, 2019.
13. Mersedeh Sadeghi, Petr Buchníček, et al. Sprint: Semantics for performant and scalable interoperability of multimodal transport. In *TRA*, 2020.
14. Marjan Hosseini, Safia Kalwar, Matteo Giovanni Rossi, and Mersedeh Sadeghi. Automated mapping for semantic-based conversion of transportation data formats. In *Int. Work. On Semantics For Transport*, volume 2447, 2019.
15. Safia Kalwar, Mersedeh Sadeghi, Alireza Javadian Sabet, Alexander Nemirovskiy, and Matteo Giovanni Rossi. Smart: Towards automated mapping between data specifications. In *The 33rd Int. Conf. on Software Engineering and Knowledge Engineering, SEKE*. KSI, 2021.
16. Nicolo Oreste Pinciroli Vago, Yuri Cossich Lavinas, Daniele C. Uchoa Maia Rodrigues, et al. INTEGRA: an open tool to support graph-based change pattern analyses in simulated football matches. In *Proc.of the 34th Int. ECMS Conf. on Modelling and Simulation*, 2020.
17. Maxim Spur, Vincent Tourre, et al. Exploring multiple and coordinated views for multilayered geospatial data in virtual reality. *Inf.*, 11(9), 2020.
18. Ricardo Langner, Ulrike Kister, and Raimund Dachselt. Multiple coordinated views at large displays for multiple users: Empirical findings on user behavior, movements, and distances. *IEEE Trans. Vis. Comput. Graph.*, 25(1), 2019.

19. Mokhtaria Hacherouf, Safia Nait Bahloul, and Christophe Cruz. Transforming xml documents to owl ontologies: A survey. *Journal of Information Science*, 2015.
20. Namyoun Choi et al. A survey on ontology mapping. *ACM Sigmod Record*, 2006.
21. Toni Rodrigues, Pedro Rosa, Jorge Cardoso, et al. Mapping xml to exiting owl ontologies. In *Int. Conf. WWW/Internet*. Citeseer, 2006.
22. Christophe Cruz and Christophe Nicolle. Rdf rules for xml data conversion to owl ontology. In *WEBIST*, 2009.
23. Chao Yin, Jianhua Gu, and Zhengxiong Hou. An ontology mapping approach based on classification with word and context similarity. In *2016 12th Int. Conf. on Semantics, Knowledge and Grids (SKG)*. IEEE, 2016.
24. Pham Thi Thu Thuy, Young-Koo Lee, and SungYoung Lee. Dtd2owl: automatic transforming xml documents into owl ontology. In *Proc. of the 2nd Int. Conf. on Interaction Sciences: Information Technology, Culture and Human*, 2009.
25. He Tan, Georges Barakat, and Vladimir Tarasov. Translating xml models into owl ontologies for interoperability of simulation systems. In *BIR Workshops*, 2015.
26. John Li. Lom: A lexicon-based ontology mapping tool. Technical report, Teknowledge Corp Palo Alto Ca, 2004.
27. Thomas R Kramer et al. Software tools for xml to owl translation. 2015.
28. Hannes Bohring and Sören Auer. Mapping xml to owl ontologies. *Marktplatz Internet: Von e-Learning bis e-Payment, 13. Leipziger Informatik-Tage*, 2015.
29. Christian Knauer, David Urbansky, Johannes Meinecke, et al. Semi-automatic semantic lifting of xml to a target ontology. In *The joint Int. symposium on natural language processing and agriculture ontology service (SNLP-AOS)*, 2011.
30. Lisa Graham. Gestalt theory in interactive media design. *Journal of Humanities & Social Sciences*, 2(1), 2008.
31. Alessio Carenini, Ugo Dell'Arciprete, Stefanos Gogos, Mohammad Mehdi Pourhashem Kallehbasti, Matteo Giovanni Rossi, and Riccardo Santoro. ST4RT–semantic transformations for rail transportation. In *TRA*, 2018.
32. Mersedeh Sadeghi, Luca Sartor, and Matteo Rossi. A semantic-based access control mechanism for distributed systems. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021.
33. Ahmad Alobaid, Daniel Garijo, María Poveda-Villalón, et al. Automating ontology engineering support activities with ontoology. *Journal of Web Semantics*, 57, 2019.
34. Mario Scrocca, Marco Comerio, Alessio Carenini, and Irene Celino. Turning transport data to comply with eu standards while enabling a multimodal transport knowledge graph. In *International Semantic Web Conference*. Springer.
35. Alireza Javadian Sabet, Matteo Rossi, Fabio A Schreiber, and Letizia Tanca. Context awareness in the travel companion of the shift2rail initiative. In *Italian Symposium on Advanced Database Systems*, 2020.
36. Alireza Javadian Sabet, Sankari Gopalakrishnan, Matteo Rossi, Fabio A. Schreiber, and Letizia Tanca. Preference mining in the travel domain. In *IEEE Int. Conf. on Artificial Intelligence and Computer Applications*, 2021.
37. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
38. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.
39. Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4), 2016.