IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 January 2023, accepted 11 March 2023, date of publication 20 March 2023, date of current version 29 March 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3259904*

RESEARCH ARTICLE

# Automated Creation of Mappings Between Data Specifications Through Linguistic and Structural Techniques

**SAFIA KALWAR[1], MATTEO ROSSI[1], AND MERSEDEH SADEGHI[2]**
[1]Politecnico di Milano, 20133 Milan, Italy
[2]Institut fur Informatik, University of Cologne, 50969 Köln, Germany

Corresponding author: Safia Kalwar (safia.kalwar@polimi.it)

This work was supported by Shift2Rail and the EU Horizon 2020 Research and Innovation Programme under Grant 826172 (SPRINT).

**ABSTRACT** The ability to perform automated conversions between different data formats is key to achieving interoperability between heterogeneous systems. Conversions require the definition of mappings between concepts of separate data specifications, which is typically a difficult and time-consuming task. In this article, we present a technique that exploits, in part, semantic web technologies to automatically suggest mappings to users based on both linguistic and structural similarities between terms of different data specifications. In addition, we show how a machine-learned linguistic model created by gathering data from domain-specific sources can help increase the accuracy of the suggested mappings. The approach has been implemented in our prototype tool, SMART (SPRINT Mapping & Annotation Recommendation Tool), and it has been validated through tests using specifications from the transportation domain.

**INDEX TERMS** Ontology, linguistic similarity, word embeddings, natural language processing, structural similarity, automated mapping.

## I. INTRODUCTION

In so-called Systems of Systems (SoSs) [1], [2], independent, heterogeneous systems built using different technologies interact to provide complex services. In the domain of SoSs, data interoperability—that is, the ability to seamlessly exchange data among systems—is a key concern [3]. However, achieving data interoperability is a challenge, as different systems typically rely on heterogeneous data specifications, but one that, crucially, needs to be tackled if one wants to combine divergent data sources into one seamless application.

Transportation systems are a paradigmatic example of this phenomenon, as they are becoming more and more integrated both at the regional (e.g., continental) and global level. Indeed, a single transport mode cannot alone meet the requirements of today's travelers, so multi-modal, integrated transportation systems have become key in modern societies. In addition, the Mobility-as-a-Service (MaaS) paradigm [4] requires the integration of the (already complex) software systems of very different Transport Service Providers (TSPs). For example, initiatives are underway in the European Union to create a Single European Transportation Area [5]. More specifically, the EU Shift2Rail Joint Undertaking [6], especially within its Innovation Programme 4, aims to provide users with a "one-stop-shop" solution that allows them to handle multi-modal trips across borders, using a single application that integrates many different services (shopping, booking, ticket issuing, etc.) from heterogeneous providers of different countries. This requires integrating software services that typically use different standards and specifications (possibly specific to a single country) to describe data such as travel offers, booking information, and so on. This leads to a great heterogeneity of data representations, which in turn significantly hinders the systems' interoperability.

These obstacles to interoperability can be overcome through the adoption of suitable conversion mechanisms between data specifications. Any conversion mechanism is usually—explicitly or implicitly—based on some mapping

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Mei.

between the concepts of the different data specifications (see, for example, the work presented in [7], which relies on technologies typical of the semantic web). In this direction, we have developed a tool, named SMART (SPRINT Mapping & Annotation Recommendation Tool [8]), whose main purpose is to automatically provide suggestions for the mapping concepts from one data specification to another. The tool takes as input two data specifications, one in XSD format and one provided as a machine-readable ontology, and produces a set of possible mappings using a two-step process: first, it looks for *linguistic* similarities between the terms appearing in the XSD-based specification and those appearing in the ontology, thus creating an initial mapping; then, it uses the *structure* of the two specifications (i.e., how terms are related to one another in each specification) to refine—and possibly extend—the initial mapping.

In this paper we build on and extend the work presented in [8] in several ways. First, we provide a **more complete and detailed account** of the mechanisms underlying the mapping generation process. Second, we improve the step that relies on the linguistic similarities between terms by exploiting a **custom, machine-learned, linguistic model** built using data retrieved from sources specific to the domain of interest (in our case, the transportation domain). The dataset that we used to train the model is itself a contribution of this work, and it is made publicly available to the research community. Finally, we perform a **systematic evaluation** of the accuracy of the suggested mappings on several case studies from the transportation domain.

In a nutshell, the paper explores the following two research questions. **Q1.** Can we create accurate, suitable mappings between different data specifications which are applicable in real-life situations? **Q2.** Can we create a domain-specific model which provides us with more targeted suggestions than a generic one?

The paper is structured as follows: Sect. II describes relevant related works and provides some background regarding the techniques used in this paper; Sect. III presents the mechanisms for generating the suggested mappings; Sect. IV describes the procedure followed to create the domain-specific model, including the gathering of the domain-specific training dataset; Sect. V presents and discusses the results of the validation, and Sect. VI concludes.

## II. BACKGROUND AND RELATED WORKS

This section provides an overview of techniques and works related to our approach. First (Section II-A) it presents work that is more loosely related to ours; then (Section II-B), it briefly describes the techniques that are at the core of the approach presented in this paper.

### A. RELATED WORKS

We consider work related to ours in the following areas: i) dealing with data heterogeneity problems in different domains, ii) data integration, and iii) word embeddings.

#### 1) DATA HETEROGENEITY

Generally speaking, integrating information from independently developed applications is difficult. Individual applications are usually not designed to cooperate and are often based on different concepts and data models [9]. Accordingly, the demand for interoperability has increased in various application domains. One such example is the healthcare domain, which has seen an increased interest in semantic interoperability [10]. Indeed, a collaboration between processes is of the utmost importance in the healthcare domain since it is a critical requirement to deliver quality service care. In [11], the authors propose an approach for the semi-automatic detection of synonyms and homonyms of process element names by measuring the similarity between business process models semantically captured through OWL specifications. This is a two-fold approach to map two different business process models semantically: it finds similarities between concepts using WordNet by looking for synonyms, and it determines their structural similarity by looking at the linguistic similarity between process attributes. This work differs from ours in terms of applied techniques, and it targets models with the same schema, whereas in our proposed approach we consider specifications with different schemas.

The transportation domain is another major application field of semantic interoperability [12], [13]. The use of ontologies enables data interoperability among different community transport service providers of vehicle-sharing services. In this case, users create and define their own ontological model in different corporate domains. As a result, it is difficult for different users and organizations to agree on one common ontological model, as this would typically not reflect the actual business requirements of each company [14].

Accordingly, the heterogeneity caused by the existence of multiple ontologies hampers systems' interoperability. For example, in [15], the authors propose an approach to adopt a collaborative inter-organizational knowledge management network methodology developed in [16]. This work, however, only targets ontologies. Therefore, to achieve ontology merging, it expects different users to agree on specific rules, which should be described specifically per ontology. The approach proposed in [17] focuses on a visualization-oriented urban mobility ontology. The ultimate goal of this work is to benefit decision-makers by providing an ontology that can support the process of developing semantically-rich visualizations with knowledge extraction and interoperability capabilities. In a similar vein, we have developed in [18] a tool that allows users to visualize the mappings generated through the mechanisms presented in this work, thus increasing the usability and transparency of the mapping tool and making it more explainable.

#### 2) DATA INTEGRATION

Regarding data integration approaches, several works target the problem of generating a global schema from a set of

heterogeneous data sources. In this direction, some approaches propose to find correspondences between elements of schemas of heterogeneous data sources. In particular, there are multiple methods in the literature for producing mappings between XML-based data sets and ontologies. For example, [19] defines rules to transform XML documents into an existing ontology, using separate rules to create mappings to OWL classes, data properties, and object properties. Many techniques propose an automatic translation of XSD into a newly-created ontology that captures the implicit semantics existing in the structure of XML documents. For example, in [20], the authors describe mechanisms to create automatic mappings from XML to RDF and from XML schema to OWL. The framework presented in [21] creates a new ontology from an XML schema and transforms instances of the XML schema into instances of the created ontology.

Our work differs from those presented above in that our aim is to map an XSD-based specification to an existing ontology rather than creating a new one; also, to achieve this, we exploit, in addition to the structure of the specifications, the linguistic properties of the terms appearing in them.

#### 3) WORD EMBEDDINGS

Several word embedding algorithms have been introduced in the literature. Word2Vec, which is used in our system and is described in Section II-B, and Glove [22] are some of the best known. In particular, Glove has been shown to produce a word vector space with a meaningful substructure, as evidenced by its state-of-the-art performance of 75% accuracy on the word analogy dataset.

ELMo (Embeddings from Language Models), introduced by Peters et al. [23], is another word embedding approach in which word vectors are learned functions of the internal states of a deep bidirectional Language Model (biLM), which is pre-trained on a large text corpus. Finally, Devlin et al. [24] propose BERT (Bidirectional Encoder Representations from Transformers), based on the bi-transformer technique, which can effectively exploit the deep semantic information of a sentence, and which can learn characterizations that can consider the context in both directions.

Even though the Word2Vec-trained models used in this work showed good performance, in the future we plan to explore alternative approaches—and in particular those mentioned above—for creating word embeddings and compare the effectiveness of the generated models against our current ones.

### B. BACKGROUND

This section first briefly describes the main features of the *Word2Vec* (W2V [25]) algorithm, which is used to train the machine-learned linguistic models at the core of the presented approach, and of the models created through it. Then, it overviews the main steps that are followed to train models with W2V, and the most relevant parameters set during the training phase.

W2V is an algorithm that, given a corpus $C$ of text—that is, a sequence of sentences—creates a model where words appearing in the corpus are associated with vectors representing their *features* (one vector per word). The idea is that words that play similar roles in the corpus have similar features; hence, in a well–trained model, their vectors are close to one another in the vector space.

Once vectors are associated with words, the notion of *cosine similarity* can be used. More precisely, given two words, their similarity can be measured through the value of the *cosine* of the angle between their corresponding vectors (see Figure 1). In particular, given vectors $\vec{a}$ and $\vec{b}$ associated with words $a$, $b$, respectively, their similarity is computed through the formula shown in Equation (1), which corresponds to the dot product of $\vec{a}$ and $\vec{b}$, normalized by the product of the vectors' norms (i.e., the absolute values of their respective lengths).

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \, ||\vec{b}||} \tag{1}$$

It is easy to see that a value of 1 for the cosine similarity between two words corresponds to the case in which the vectors are the same, which in turn entails that the words are equivalent in the corpus. Conversely, a value of 0—in which case the vectors are orthogonal—signals that the two words are unrelated to one another. Figure 1 provides a graphical depiction (in a 2D feature space) of the notion of cosine similarity between words.

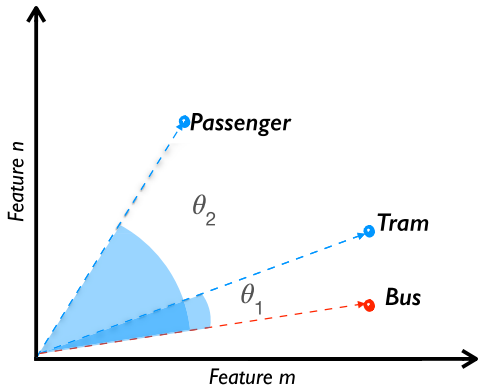


**FIGURE 1.** Cosine similarity using W2V.

More precisely, words *Bus* and *Tram*, which have a similar meaning, would have similar vectors. Hence the value of their cosine similarity would be close to 1; the vectors for *Passenger* and *Bus*, on the other hand, would be farther apart, and the value of their cosine similarity would be lower.

The W2V technique is able to associate feature vectors not only with single words but also with "phrases", which are expressions made of multiple words, for example "train station", or "New York". In particular, we refer to phrases made of 2 words as "bi-grams" and, in general, to phrases made of multiple words as "n-grams". W2V associates

**Sentence: the pink horse is eating**

| Sentence | Word pairs |
|---|---|
| **the** pink horse is eating | (**the**, pink), (**the**, horse) |
| the **pink** horse is eating | (**pink**, the), (**pink**, horse), (**pink**, is) |
| the pink **horse** is eating | (**horse**, the), (**horse**, pink), (**horse**, is), (**horse**, eating) |
| the pink horse **is** eating | (**is**, pink), (**is**, horse), (**is**, eating) |
| the pink horse is **eating** | (**eating**, horse), (**eating**, is) |

**FIGURE 2. Example of the impact of the value of parameter *window_size* on the identification of the pairs of words during the training of W2V models.**

vectors directly with phrases when these appear very frequently in the corpus (with respect to the separate composing words). Even when W2V cannot directly associate a feature vector with a sequence of words, it can compute, given the vectors of the composing words, a vector for the whole sequence. In a well-trained model, vectors resulting from the combination of words (e.g., ''vehicle'' and ''journey'') will be close to vectors for similar words (e.g., ''trip''). Typically, the combination consists of adding the composing vectors, though other possibilities (e.g., vector average) are available.

Let us now provide an overview of the steps that are taken when training a model using *W2V* and of the most relevant training parameters that influence the resulting model. First, the raw data (the corpus, i.e., a sequence of sentences) is collected and pre-processed. Pre-processing includes the cleaning of the data (e.g., removing spurious characters), lowercasing, lemmatization (i.e., replacing a word with its base form, for example in the case of verbs), and removing punctuation and numerical forms. In addition, words that appear only very rarely are removed to avoid introducing noise in the model. The so-created corpus is further processed and fed to the training algorithm to create the final model. Several implementations of the W2V technique are available in the literature (see, e.g., [26]); in this work, we use the Gensim library [27], which also offers a few useful additional utilities. Before performing the actual training, a step to identify phrases (bi-grams, etc.) in the corpus is typically performed (e.g., through module *gensim.models.phrases* of the Gensim library). In particular, each pair of consecutive words is analyzed and—provided the pair appears at least *min_count* times in the corpus, where *min_count* is a parameter set by the user—it is assigned a score that measures how many times the pair appears with respect to the individual words (pairs that appear more often than individual words, hence that are more relevant in pair than individually, have higher scores). If the score is greater than the value of user-defined parameter *threshold*, the pair is considered a phrase; otherwise it is not (only individual words are considered). Notice that the phrase identification step is performed in multiple iterations to identify also phrases that are made of more than two words. As mentioned above, after phrases are identified in the corpus (e.g., ''railway station''), they are treated as single terms, and the algorithm will associate with each of them a single feature vector. As a final processing step before training, all terms (words or phrases) that appear less than *min_count* times in the corpus are removed. Finally, the actual training can start.

To train the model, W2V analyzes pairs of words that are close to one another, where ''close'' is determined by parameter *window_size*. For example, a value of 2 for *window_size* implies that to determine the set of pairs of words to be analyzed, for every word $w$ appearing in the corpus, every word that is at a distance of at most 2 (either before or after $w$) is paired with $w$, as depicted in Fig. 2 (where each row corresponds to a different $w$, highlighted in bold). Notice that the actual distance between the words in a pair does not matter, only that the words are inside the stated window. A smaller window size gives results that are more syntactic in nature, whereas a larger window (e.g., *window_size* $>$ 5) produces results that are more semantic. Larger values of parameter *window_size* increase the training time as more pairs are analyzed in training. The parameter *vector_size*, instead, determines the number of dimensions in the feature vector associated with each term in the model. The number of dimensions is typically chosen also depending on the size of the corpus, and the higher the number of dimensions, the longer the training time. A typical number of dimensions, which seems to provide a good trade-off between accuracy and training time, is 300; however, one could experiment with different sizes of the feature vector, especially if the size of the corpus exceeds 100M words. As mentioned above, during the training phase W2V considers pairs of words that appear in windows of size *window_size*. However, as also highlighted by the example in Figure 2, many of these pairs feature frequent but not very meaningful words such as articles and prepositions (e.g., ''**the**, pink''). To reduce the impact of these words on the training, W2V allows users to ''down-sample'' them—i.e., to consider only some of their instances, but not all. In particular, the parameter *sample* determines the probability that each instance is considered in the training phase depending on the frequency of each word in the corpus. The value of parameter *sample*, which typically ranges between 1E-3 and 1E-6, has a significant impact on the outcome of the training, as also shown in Section V.

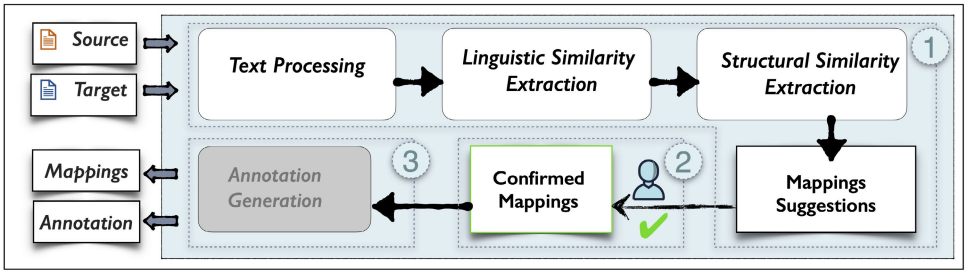


**FIGURE 3.** Workflow of the SMART tool.

## III. METHODOLOGY

The overall workflow for creating the mappings, implemented in the SMART tool, is shown in Figure 3. The process starts by taking two data specifications, referred to as "source" and "target"[1] as input. The procedure is composed of three main steps. The first step involves creating mappings by identifying similar terms for the given data specifications. In particular, for each term in the source specification, SMART suggests three terms from the target standard, along with a confidence score. As the name suggests, it represents how the system is confident about the similarity of the two terms, which is an indicator of the similarity score calculated through the formula (1). The second step then allows the user to check and confirm those mappings. Users can either manually inspect the suggestions or let SMART automatically choose the suggestions with the highest confidence score. Finally, in the third step, the confirmed mappings are further used to generate annotations that capture the mappings and that can be used to automatically translate data conforming to one specification into the other format [7].

In this paper, we focus on the first phase of the workflow, which deals with the automated generation of the mappings; the remaining steps are presented in [8].

The mechanism to automatically generate suggested mappings relies on two main techniques: *(i)* linguistic mapping (which is detailed in Section III-A), and *(ii)* structural mapping (described in Section III-B). The former applies Natural Language Processing (NLP) and Machine Learning (ML) techniques to identify similar terms. The latter, instead, further exploits the structure of the data in the source and target specifications to derive a new and more precise set of mappings. The overall mapping procedure is captured by Algorithm 1. For clarity, some parts have been encapsulated in sub-algorithms that are shown as algorithms 2, 3 and 4.

**Algorithm 1** Mapping Algorithm

```
1: procedure SMARTMapping
2: input: X: XSD file, O: OWL file
3: output P: set of triples ⟨xt, ot, s⟩, xt ∈ X, ot ∈ O, s:Confidence score
4:     oCl ← O.Class
5:     oObPr ← O.ObjectProperty
6:     oDtPr ← O.DatatypeProperty
7:     xName ← (X.Attribute.name ∪ X.Element.name)
8:     xType ← (X.Attribute.type ∪ X.Element.type)
9:     xCl ← X.ComplexTypes
10:    xObPr ← {xb|xb ∈ xName if XType(xName) = ComplexType}
11:    xDtPr ← {xd|xd ∈ xName if XType(xName) = Datatype}
12:
13:    ▷ Create initial mapping between terms using linguistic similarity
14:    mappedClass ← W2VlinguisticMap(xCl, oCl)
15:    mappedObjProp ← W2VlinguisticMap(xObPr, oObPr)
16:    mappedDataProp ← W2VlinguisticMap(xDtPr, oDtPr)
17:
18:    ▷ New mappings between object properties (Alg. 2)
19:    mappedObjProp ← AddObjPropFromClasses(
                           mappedObjProp, mappedClass,
                               xObPr, oObPr )
20:
21:    ▷ New mappings btw. classes based on classes & obj. prop. (Alg. 3)
22:    mappedClass ← AddClassesFromClassesAndObjProp(
                           mappedObjProp, mappedClass,
                               xCl, oCl )
23:
24:    ▷ New mappings between classes based only on properties (Alg. 4)
25:    mappedClass ← AddClassesFromObjProp(
                           mappedObjProp, mappedClass,
                               xCl, oCl )
26:
27:    return mappedClass ∪ mappedObjProp ∪ mappedDataProp
28: end procedure
```

### A. LINGUISTIC MAPPING PHASE

In the first phase of the mechanism (corresponding to lines 7-16 of Algorithm 1), a W2V-trained model is used to create a first set of suggested mappings between terms of the two input data specifications. For this reason, we refer to this as the "linguistic step" of the process, though, as it will be apparent later, the structure of the specifications is also exploited in some points. In this work, we use and experiment with several W2V-trained models; in particular, we use a general-purpose model (referred to, in the following, as the *Google News* model), publicly available from the literature [28] and which has been trained on generic documents, and several domain-specific models that we trained ourselves on transportation-related documents (see Section IV for further details). The linguistic mapping phase consists of several steps, which are detailed below.

[1]In the rest of this paper, we refer to the input specifications as "source" and "target" to distinguish them. However, these notions are interchangeable, and there is no direction in the mapping creation.

**TABLE 1. XSD to OWL structural mapping rules.**

| XSD | OWL Type and Name |
|---|---|
| <xsd:complexType name = **"A"**><br><xsd:complexContent><br><xsd:extension base = **"B"**><br>*Where B is another ComplexType* | Class(**A**)<br>SubClassof(**B**) |
| <xsd:complexType name = **"A"**><br><xsd:complexContent><br><xsd:extension><br><xsd:element name = **"E1"** type= **"B"**><br>*Where B is another ComplexType* | ObjectProperty(**hasE1**)<br>Domain(Class(**A**))<br>Range(Class(**B**)) |
| <xsd:complexType name = **"A"**><br><xsd:complexContent><br><xsd:extension><br><xsd:element name = **"E1"** type = **"D"**><br>*Where D is a DataType* | DataTypeProperty(**hasE1**)<br>Domain(Class(**A**))<br>Range(DataType(**D**)) |
| <xsd:complexType name = **"A"**><br><xsd:complexContent><br><xsd:extension><br><xsd:attribute name = **"Attr1"** type = **"B"**><br>*Where B is another ComplexType* | ObjectProperty(**hasAttr1**)<br>Domain(Class(**A**))<br>Range(Class(**B**)) |
| <xsd: complexType name = **"A"**><br><attribute name = **"Attr1"** type = **"D"**><br>*Where D is a DataType* | DataTypeProperty(**hasAttr1**)<br>Domain(Class(**A**))<br><br>Range(DataType(**D**)) |

### 1) SPECIFICATION PRE-PROCESSING

In the first step, the input data specifications are read, and the terms appearing in them are organized into several categories depending on their definitions. More precisely, the algorithm assumes that one specification is in XSD format while the other is an ontology defined in OWL format (though, as explained later in this section, the mechanism also works when both inputs are in XSD or OWL format). The categorization of terms is performed according to the following rules, which are also captured by Table 1.

We assume that the XSD specification [29] describes concepts as *ComplexTypes* that can have *Elements* and *Attributes*, which in turn have types that can be *Datatypes* or other *ComplexTypes*. The OWL specification [30], instead, represents knowledge as a set of *Classes* to which properties (which can be *Datatype Properties* or *Object Properties*) belong. Properties correspond to binary relations between their domain (which are instances of *Classes*) and their ranges (which are instances of *Classes* or of *Datatypes*). The algorithm first iterates through the terms of the OWL file. It creates three sets, *oCl*, *oObPr* and *oDtPr*, containing, respectively, classes, object properties, and datatype properties (lines 4-6 of Algorithm 1). Then, it goes through the terms of the XSD file (lines 7-16), and it builds three further sets of terms: *xCl*, which are candidates to be mapped to OWL classes; *xObPr*, which are candidates to be mapped to OWL object properties; and *xDtPr*, which candidates to be mapped to OWL datatype properties. To do this, it applies the rules of Table 1, which are briefly described in the following.

The rule captured by the first row of Table 1 defines that a *ComplexType* (say, "A") defined in the XSD file should correspond to a *Class* in the OWL file. In addition, if the *ComplexType* extends another *ComplexType* (say, "B"), then this should correspond to a *SubClass* relationship in the OWL file. The rule in the second row defines that an element (say, "E1") of a *ComplexType* ("A") that has as a type another *ComplexType* ("B") should correspond to an *ObjectProperty* in the OWL file; if instead, the type of the element is a *DataType* ("D"), then it should correspond to a *DataProperty* (third row). Similarly (fourth row), if an attribute (say, "Attr1") of *ComplexType* "A" has as a type another *ComplexType* "B", it should correspond to an *ObjectProperty* in the OWL file; if instead (fifth row) the type of the attribute is a *DataType* ("D"), it should correspond to a *DataProperty*.

### 2) SIMILAR TERM RETRIEVAL

In the second step, the W2V-trained model is used to retrieve, for each term appearing in each of the aforementioned six sets (*oCl*, *xCl*, etc.), the $n$ most similar terms (see Section II-B), where $n$ is a configurable parameter.[2] Then, if $x$ (resp., $y$) is the number of terms from the source (resp., target) standard, after getting $n$ similar terms from the W2V-trained model, we obtain two matrices (one for the source specification and one for the target specification) that contain $x \cdot (n+1)$ and $y \cdot (n+1)$ elements, respectively. Table 2 shows a snippet of examples of tables (indicated as MatS for the source specification and MatT for the target one) obtained at the end of this step (with $n = 4$), where the left-most column shows the original terms from the source (TermS) and target (TermT) standards, while the other columns (SimTermS1, SimTermS2, etc.) show, for each original term, the words suggested by the W2V-trained model.

**TABLE 2. Matrix representation after step (b).**

| MatS: W2V suggestions for terms in the first specification | | | | |
|---|---|---|---|---|
| **TermS** | **SimTermS1** | **SimTermS2** | **SimTermS3** | **SimTermS4** |
| *RequestStop* | Request | Stop | Pick_Drop | Wait |
| *FareZone* | Fare | Zone | Zonal | TravelCard |
| **MatT: W2V suggestions for terms in the second specification** | | | | |
| **TermT** | **SimTermT1** | **SimTermT2** | **SimTermT3** | **SimTermT4** |
| *Layover* | Stop | Schedule | Stopover | Waiting_Time |
| *AccessZone* | Access | Zonal | Restrict | Accessibility |

### 3) TERM MATCHING

In this step, the algorithm retrieves the W2V-computed similarity of each pair of terms (one taken from MatS and one from MatT) appearing in the matrices obtained at the end of step (b). The result is a list of triples $\langle$SimTermSi, SimTermTj, $Sim_{ij}\rangle$, where $Sim_{ij}$ is the cosine similarity of SimTermSi and SimTermTj. Notice that the

[2]We experimented with different values for parameter $n$ (3, 5, 10, 20), and in the experiments of Section V we finally settled on $n = 10$.

algorithm only considers pairs ⟨SimTermSi, SimTermTj⟩ that belong to rows corresponding to terms (TermS and TermT) that have the same nature, as determined by step (a) (i.e., such that TermS belongs to *xCl* and TermT belongs to *oCl*, or to *xObPr* and *oObPr*, or to *xDtPr* and *oDtPr*, respectively).

We filter out the triples whose similarity value $Sim_{ij}$ is lower than a threshold *th*, where *th* is a configurable parameter set to 0.5 in our experiments. We consider that each pair of terms whose similarity is higher than the threshold is a potential generator of a mapping, according to the rules of the next step (d).

#### 4) SUGGESTION CONSOLIDATION

In the final step of the linguistic phase, each triple ⟨SimTermSi, SimTermTj, $Sim_{ij}$⟩ that is left after the final filtering performed in step (c) is traced back to the pair of original terms ⟨TermS, TermT⟩. Also, for each pair of original terms ⟨TermS, TermT⟩, we count the number of triples ⟨SimTermSi, SimTermTj, $Sim_{ij}$⟩ that are traced back to it. For example, consider Table 2 again and imagine that, after step (c), we have a triple ⟨Request, Stop, 0.677⟩. We trace it back to the original pair of terms ⟨RequestStop, Layover⟩, and we increase by one the counter the number of matches between RequestStop and Layover. Finally, we compute the confidence score $CS_{ts,tt}$ for the mapping ⟨TermS, TermT⟩ as the average of the values $Sim_{ij}$ of the triples ⟨SimTermSi, SimTermTj, $Sim_{ij}$⟩ that trace back to it. At the end of this step, we produce a set of triples ⟨TermS, TermT, $CS_{ts,tt}$⟩.

Notice that steps (b)-(d) described above are encapsulated in lines 14-16 of Algorithm 1, and they are separately applied depending on the nature (class, object property, data property) of each original term.

### B. STRUCTURAL MAPPING PHASE

In the second phase of Algorithm 1, captured by lines 19-25, we use the structure of the ontology as guidance to further refine the mappings returned by the linguistic mapping phase. The results produced by the linguistic mapping are stored in three sets of triples of the form ⟨TermS, TermT, $CS_{ts,tt}$⟩ named *mappedClass*, *mappedObjProp* and *mappedDataProp*, where TermS and TermT are either both names of *Classes*, or of *ObjectProperties*, or of *DataProperties*, depending on the set. Notice that, for the sake of structural mapping, we categorize terms appearing in XSD specifications as *Classes*, *ObjectProperties*, and *DataProperties* according to the rules of Table 1. Hence, in this step, we consider that each specification defines triples of the form ⟨Domain, ObjectProperty, Range⟩. In the following, we indicate a triple from the source (resp., target) specification as ⟨DomainS, ObjectPropertyS, RangeS⟩ (resp., ⟨DomainT, ObjectPropertyT, RangeT⟩).

Then, in the structural mapping phase, as depicted in Figure 4, we look at pairs of triples, ⟨DomainS, ObjectPropertyS,RangeS⟩ and ⟨DomainT, ObjectPropertyT, RangeT⟩, that have some elements already mapped to one another, and use this information to suggest new mappings. These mechanisms are captured by algorithms 2-4 referenced in Algorithm 1, and are detailed in the rest of this section. The algorithms differ depending on which parts of the triples are already matched to one another.

**Algorithm 2** New Object Properties

```
1:  procedure AddObjPropFromClasses
2:  input: mappedObjProp, mappedClass, xObPr, oObPr
3:  output P: set of triples ⟨xt, ot, s⟩, xt ∈ X, ot ∈ O, s:Confidence score
4:
5:      propList ← ∅
6:      foreach (xc_j, oc_j, s_j) ∈ mappedClass do
7:          foreach (xc_i, oc_i, s_i) ∈ mappedClass do
8:              foreach (xp, op) ∈ xObPr × oObPr do
9:                  if xc_i = xp.ComplexType & oc_i = op.Domain &
10:                    xc_j = xp.Type & oc_j = op.Range then
11:                     s ← (s_i + s_j)/2
12:                     propList ← propList ∪ {(xp, op, s)}
13:                 end if
14:             end foreach
15:         end foreach
16:     end foreach
17:     return mappedObjProp ∪ propList
18: end procedure
```

#### 1) SUGGEST PROPERTIES IF DOMAINS AND RANGES MATCH

If in the data specifications, there are two triples ⟨DomainS, ObjectPropertyS, RangeS⟩ and ⟨DomainT, ObjectPropertyT, RangeT⟩ such that, in set *mappedClass*, DomainS is mapped to DomainT and RangeS is mapped to RangeT, then triple ⟨ObjectPropertyS, ObjectPropertyT, $CS_{ops,opt}$⟩ is added to set *mappedObjProp*, where $CS_{ops,opt}$ is the average of the confidence scores of the mappings between domains and ranges—i.e., pair ⟨ObjectPropertyS, ObjectPropertyT⟩ is suggested with a confidence score $CS_{ops,opt}$. This step is performed by the procedure invoked at line 19 of Algorithm 1; in particular, the addition of every single new pair is performed by lines 9-13 of Algorithm 2.

#### 2) SUGGEST DOMAINS (RESP., RANGES) IF PROPERTIES AND RANGES (RESP., DOMAINS) MATCH

If ObjectPropertyS is mapped to ObjectPropertyT in *mappedObjProp* and RangeS is mapped to RangeT in *mappedClass*, then pair ⟨DomainS, DomainT⟩ is suggested with confidence score $CS_{ds,dt}$, where $CS_{ds,dt}$ is the average of the confidence scores of the mappings between properties and ranges. Similarly, if ObjectPropertyS is mapped to ObjectPropertyT and DomainS is mapped to DomainT, then we suggest pair ⟨RangeS, RangeT⟩. This step is performed by the procedure invoked at line 22 of Algorithm 1, which is detailed in Algorithm 3.

#### 3) SUGGEST DOMAINS AND RANGES IF PROPRIETIES MATCH

In this case, if ObjectPropertyS is mapped to ObjectPropertyT in *mappedObjProp*, we suggest pairs ⟨DomainS, DomainT⟩ and ⟨RangeS, RangeT⟩, both with confidence score that

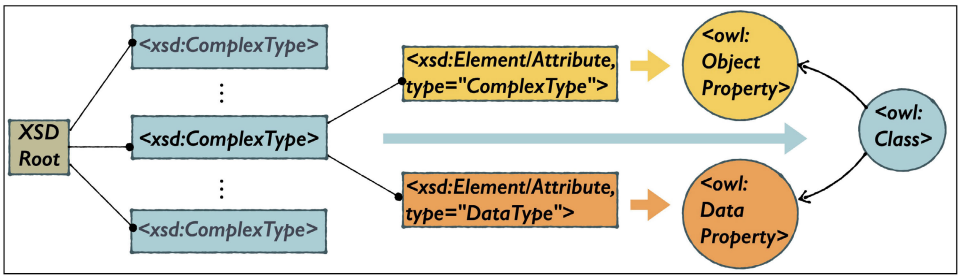


**FIGURE 4.** XSD to OWL structural mapping.

**Algorithm 3** New Classes Based on Properties and Classes

```
1: procedure AddClassesFromClassesAndObjProp
2: input: mappedObjProp, mappedClass, xCl, oCl
3: output P: set of triples ⟨xt, ot, s⟩, xt ∈ X, ot ∈ O, s:Confidence score
4:
5:     rangeList, domainList ← ∅
6:     foreach (xc_i, oc_i, s_i) ∈ mappedClass do
7:         foreach (xp_j, op_j, s_j) ∈ mappedObjProp do
8:             foreach (xc, oc) ∈ xCl × oCl do
9:                 ▷ Domain and property are mapped, we map range
10:                if xc_i = xp_j.ComplexType & xc = xp_j.Type &
11:                  oc_i = op_j.Domain & oc = op_j.Range then
12:                    s ← (s_i + s_j)/2
13:                    rangeList ← rangeList ∪ {(xc, oc, s)}
14:                end if
15:                ▷ Property and range are mapped, we map domain
16:                if xc_i = xp_j.Type & xc = xp_j.ComplexType &
17:                  oc_i = op_j.Range & oc = op_j.Domain then
18:                    s ← (s_i + s_j)/2
19:                    domainList ← domainList ∪ {(xc, oc, s)}
20:                end if
21:            end foreach
22:        end foreach
23:    end foreach
24:    return mappedClass ∪ rangeList ∪ domainList
25: end procedure
```

**Algorithm 4** New Classes Based on Properties

```
1: procedure AddClassesFromObjProp
2: input: mappedObjProp, mappedClass, xCl, oCl
3: output P: set of triples ⟨xt, ot, s⟩, xt ∈ X, ot ∈ O, s:Confidence score
4:
5:     rangeList, domainList ← ∅
6:     foreach (xp, op, s) ∈ mappedObjProp do
7:         foreach (xc_i, oc_i) ∈ xCl × oCl do
8:             foreach (xc_j, oc_j) ∈ xCl × oCl do
9:                 if xc_i = xp.ComplexType & oc_i = op.Domain &
10:                  xc_j = xp.Type & oc_j = op.Range then
11:                    s ← (s ∗ 0.6)
12:                    domainList ← domainList ∪ {(xc_i, oc_i, s)}
13:                    rangeList ← rangeList ∪ {(xc_j, oc_j, s)}
14:                end if
15:            end foreach
16:        end foreach
17:    end foreach
18:    return mappedClass ∪ rangeList ∪ domainList
19: end procedure
```

is 60% that of the mapping between the properties (i.e., that is equal to $0.6 \cdot CS_{ops,opt}$). This step is performed by the procedure invoked at line 25 of Alg. 1 (see also Alg. 4). The rationale behind the choice of the 0.6 factor is the following. Consider the case in which triples ⟨DomainS, ObjectPropertyS, RangeS⟩ and ⟨DomainT, ObjectPropertyT, RangeT⟩ is such that properties ObjectPropertyS and ObjectPropertyT are matched with confidence score $c_p$, but domains and ranges are not mapped. If we consider that the domains are matched with confidence score $c_d = 0$, and we suggest the matching of the ranges RangeS and RangeT, by computing, as done above, the average of $c_p$ and $c_d$ to determine the confidence score of the new mapping, we would obtain the value $c_r = (c_p + 0)/2 = 0.5\ c_p$. However, since the new mapping ⟨RangeS, RangeT⟩ arises from structural considerations and not only linguistic ones, we consider this as grounds to increase the confidence score of the mapping, and we choose 0.6 as a consequence.

Although the algorithms presented above assume that one specification is given as an XSD file and the other as an OWL ontology, they have been adapted to also work when the input specifications have the same format (i.e., they are both XSD files or both ontologies). More precisely, if both inputs are XSD files, then the algorithm performs the same pre-processing step explained in point (a) of the linguistic mapping on both files to extract a set of "candidate classes" $xCl_1, xCl_2$ from each file, which is then used in the rest of the algorithm instead of $xCl$ and $oCl$ (similarly for object and data properties).

## IV. CREATING DOMAIN-SPECIFIC MODELS

In this section, we explain how we trained a domain-specific model using W2V, to improve the quality of automatically created suggestion of our framework.

The overall workflow is depicted in Figure 5. In particular, we first collected a set of suitable data sources; then, the texts retrieved from each data source were pre-processed and cleaned to produce text that was suitable to be fed to the W2V algorithm, where the cleaning process is specific for each data source. Section IV-A describes the selected data sources and the corresponding pre-processing operations, whereas Section IV-B provides some details regarding the training procedure and, in particular, the values of the parameters used for the training, and the produced models.

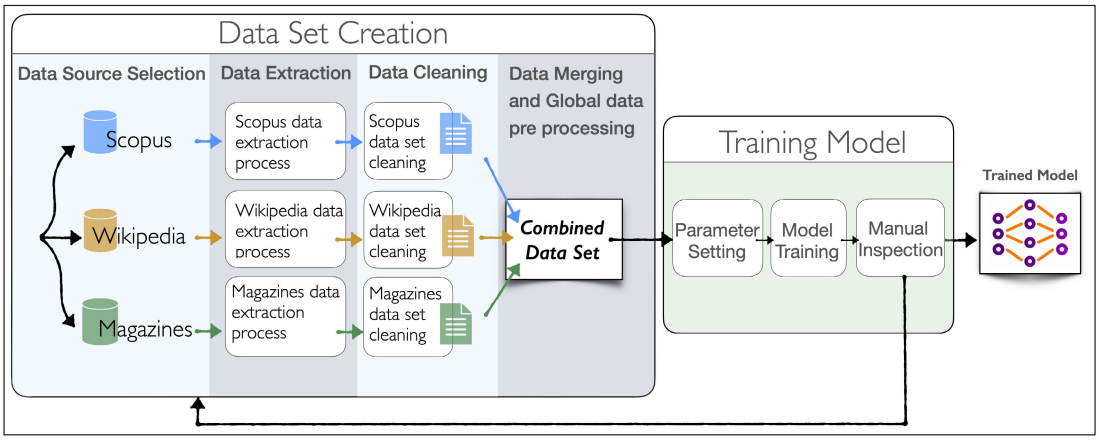


**FIGURE 5.** Model training workflow.

### A. CREATING THE INPUT DATASET

As mentioned in Section II-B, the input to a W2V training procedure is a dataset that consists of a sequence of paragraphs, where each paragraph is a list of words. The dataset should have a few desired characteristics, as described below. First of all, each paragraph should have a cohesive meaning. Also, the dataset should include as many domain-specific (e.g., transportation) terms as possible. Finally, the paragraphs in the dataset should be "clean"—i.e., they need to be in a format that can be understood by the training algorithm, for example they should not contain spurious characters. Hence, selecting the right sources is crucial. For example, data sources that provide meaningful data but that are very difficult to put in a format that is amenable for training are not very useful—consider, for instance, paper documents, which would require a complex acquisition and cleaning process of scanning, text recognition, etc. We identified three interesting data sources that have a good trade-off between pertinence and usability: (i) abstracts from scientific articles about transportation from the Scopus database; (ii) Wikipedia pages talking about transportation issues; (iii) articles in electronic magazines about transportation. In the rest of this section, we present the most relevant characteristics of each data source and the specific steps that we had to take to prepare their data for the training phase. Finally, we describe how the data from the three sources were combined into the corpus fed to the W2V algorithm.

#### 1) DATA COLLECTION FROM SCOUPS

Scientific articles tend to be focused and cohesive by their very nature. In addition, the publication outlet (the specific conference or journal) and the keywords that are included with each paper facilitate the task of recognizing texts that are closely related to the desired domain. On the other hand, the main text of scientific articles usually includes technical terms that are of little interest to our purposes. To mitigate this issue, we retrieve only the paper *abstracts*, which normally provide higher-level, less technical descriptions of the work (hence they include more general terms, which are of greater interest for the corpus being built). There exist multiple repositories of scientific articles, such as IEEE Xplore,[3] the ACM Digital Library,[4] and Springer Link.[5] We finally opted for the Scopus repository,[6] because it is more diverse and general than those mentioned above and it aggregates work from different sources and domains. Abstracts are retrieved according to expressions that appear in the text or in the list of keywords, which should be representative of the desired domain. In particular, for this work, we have gathered a set of relevant expressions composed of the most common terms usually appearing in transportation domain sources. Such expressions include "Public Transport", "Electric Scooters", "Street traffic control", "Urban Transportation", "Sustainable mobility", "Highway Traffic Control". Ultimately, we retrieved around 1000 articles from Scopus, which underwent some pre-processing and cleaning steps before being merged into one final file. In particular, the data fetched from Scopus is downloaded as a single CSV (Comma-Separated Values) file, which already contains a list of paragraphs (one per row, where each paragraph corresponds to an abstract), as required by the W2V algorithm. The cleaning step involves removing special (non-alphabetic) characters, extra spaces, and empty lines.

#### 2) DATA COLLECTION FROM WIKIPEDIA

In this case, we build the dataset by retrieving relevant pages from Wikipedia. As before, the search is done through significant keywords that are manually selected from the various resources in the domain of interest, including domain-specific standards and data specifications (for example, those mentioned in Section V). The keywords are organized in a list (the "term query list") that is used

[3] ieeexplore.ieee.org
[4] dl.acm.org
[5] link.springer.com
[6] www.scopus.com

to automatically retrieve the pages. In this work, we used transportation-related keywords such as ''Bus Station'', ''Location'', ''Passenger'', ''Quay'', ''Airport''. We retrieve the corresponding Wikipedia page for each element in the term query list, which typically includes links to other related pages. We follow these links and, to avoid retrieving pages that are only indirectly related to the starting one, we stop at the first level of linking. Data fetched from Wikipedia is already in text format, hence, as in the case of Scopus abstracts, the data cleaning concerns the removal of special characters, empty spaces/lines, and unwanted punctuation. Finally, the cleaned data is ready to be merged into the final corpus.

#### 3) DATA COLLECTION FROM TRANSPORTATION MAGAZINES

The third data source used to create the training corpus are electronic documents from books and magazines focusing on the desired domain. We selected relevant magazines through suggestions provided by experts in the transportation domain, for example, Intelligent Transport [31], Metro [32], and others. These magazines cover topics such as ''Smart Cities'', ''Metro and Light Rail'' and ''Road Traffic Congestion''. We also performed a manual internet search to retrieve books and white papers tackling issues mentioned in the magazine articles (e.g., [33], [34]). The data fetched from magazines and books is in pdf format rather than pure text, and a pre-processing step is necessary to make it suitable to be integrated in the final corpus. This was done through a script that, given a pdf document, traverses it and breaks it down into paragraphs, where each paragraph corresponds to one page in the pdf document. Then, a cleaning step is performed similar to those carried out for the other sources, and finally, the dataset is merged with the others for the training step.

In the final stage before launching the training, data are combined from the three sources into a single file, as they in are the same format after cleaning. The final version of the corpus (obtained through various iterations, as described in Section IV-B) has a size of 67MB, where 45MB of data originate from magazines and books, 8MB from Wikipedia pages, and the remaining 14MB from abstracts retrieved from Scopus. Then, to make sure that the dataset does not contain spurious elements, further cleaning steps are performed. More precisely, special characters, punctuation symbols (e.g., commas) and any remaining empty lines are removed. Also, duplicated data and filler words (e.g., ''a'', ''an'', ''the'', ''of'') are deleted. After this, the dataset is ready for the training phase. The corpus used to train the W2V model is available at [35].

### B. TRAINING THE MODEL

The goal of the training phase is to obtain a model that is of good quality, which means that it is able to provide reasonable similarity measures between words (including n-grams). The quality of the model has a deep impact on the ability of the whole mapping creation process to provide good suggestions; hence, we can say that, ultimately, it manifests itself in the ability to provide accurate mappings, which is evaluated in Section V. The quality of the trained model is influenced both by the corpus (size, variety) provided as input to the training algorithm, and by the parameters used during the training phase. Hence, the creation of the dataset and the corresponding training are two tightly related processes that mutually influence one another. As depicted in Figure 5, we followed an iterative process where after each iteration, the quality of the trained model was analyzed through manual inspection and, if it was deemed unsatisfactory, the dataset was expanded, and the model was re-trained, until the quality was deemed sufficient. In addition, during this corpus-building step, we experimented with various values, for the training parameters (see Section II-B), until we settled on a final domain-specific trained model.

To evaluate the goodness of the trained model, we manually inspected it using a qualitative approach (notice that it is not possible to use a quantitative, automated approach to evaluate the goodness of the trained model, as there is no ''ground truth''—i.e., a reference model—against which the trained model can be compared). More precisely, we selected a few representative words and n-grams of the target domain (in this case, transportation), for example ''rail station'', ''path'', ''stop'', also taking inspiration from the terms appearing in standards widely used in the domain. Then, we used the trained model to retrieve the terms that have the highest similarity to the selected phrases, and we manually checked them. When the set of similar words was deemed reasonable, we considered the trained model satisfactory. We refer to the final, domain-specific trained model as the *Transport model*.

The size of the final Transport model was 33MB. Table 3 shows the number of phrases that have a vector associated with them in the Transport model, compared against the one in the pre-trained model based on the Google News corpus. As the table shows, the Transport model contains a much smaller number of phrases (and it takes only a few minutes to train on commodity hardware, as depicted in Table 4). Nevertheless, as Section V shows, its performance in terms of the accuracy of suggested mappings is overall better than the one of the Google News model. Concerning the values of the hyper-parameters used in the training phase, after experimenting with various values it turned out that the default values (a *window_size* of 5, a *vector_size* of 300, and a *min_count* of 2) worked very well, so we used them. The only parameter that had a significant impact on the outcome

**TABLE 3. Number of words/phrases in domain-specific and general W2V-trained models.**

| Model | Number of words/phrases in model |
|---|---|
| Transport model | 12282 |
| Google news | 3000000 |

**TABLE 4. Transport model training time depending on the value of parameter *sample*. Training was carried out on a MacBook Air, with an Intel® 1.6GHz i5 processor and 8GB of RAM.**

| Sample value | Training Time |
|---|---|
| 1E-6 | 2.55 minutes |
| 2E-6 | 2.94 minutes |
| 2E-4 | 7.16 minutes |

of the training was the *sample* value (see Section II-B). We experimented with many different values of *sample*, varying from 1E-6 to 1E-2, for a total of 17 different trained models. Of these 17 models, 3, corresponding to *sample* values 1E-6, 2E-6, and 2E-4, seemed the most promising. These models were finally selected to carry out the evaluation presented in the next section (the trained models are available at [35]).

## V. VALIDATION

This section presents the experiments that we carried out to provide an answer to the research questions introduced in Section I, which are repeated here for the ease of the reader:

- Q1: Can we create accurate, suitable mappings between different data specifications which are applicable in real-life situations?
- Q2: Can we create a domain-specific model which provides us with more targeted suggestions than a generic one?

In this evaluation, we focused on data specifications from the transportation domain; in particular, we selected several XSD-based and ontology-based specifications, and we ran our tool on various pairs of them. The rest of this section first describes the selected data specifications (Section V-A), then it presents the methodology used to evaluate the algorithm (Section V-B) and the corresponding results (Section V-C); finally, Section V-D discusses the obtained results.

### A. SELECTION OF THE DATA SPECIFICATIONS

In selecting the data specifications from the transportation domain to be used for the validation of the automated mapping generation mechanisms, we considered a few desired characteristics. In particular, we required that the specifications be used by key transportation actors and that they cover a wide range of terms used in public transportation services or models. Finally, we selected three XSD-based specifications—the Network Timetable Exchange (NeTEx, [36]), the Full Service Model (FSM, [37]) and the Traveler Realtime Information and Advisory Standard (Trias, [38])—and two ontology-based specifications—the Public Transport Reference Data Model (Transmodel, [39]) and the one developed within the Information Technologies for Shift2Rail (IT2Rail, [6]) project.

Table 5 lists, for each selected specification, the number of unique terms appearing in it. It also shows how many of them also appear in the custom-trained Transport model and in the generic Google News model. In the rest of this section, we provide a brief overview of each chosen specification.

**TABLE 5. For each specification, number of unique terms and number of terms appearing in W2V-trained models.**

| Standards | Unique Terms | Transport model | GoogleNews model |
|---|---|---|---|
| Transmodel | 231 | 164 | 209 |
| IT2Rail | 543 | 426 | 481 |
| NeTEx | 174 | 131 | 151 |
| FSM | 113 | 101 | 113 |
| Trias | 128 | 113 | 126 |

**Transmodel** is a European Committee for Standardization (CEN) reference data model for public transportation information [40], which provides an abstract model of common public transport concepts related to various areas, including timetabling, fares, operational management, real-time data, vehicle scheduling, network description, timing information, and journey planning. Transmodel can support the development of software applications and their interaction or combination in an integrated information system. In our validation experiments, we used an ontological version of the Transmodel standard, which is publicly available [39] and covers a subset of the topics captured by the original standard, such as journeys, facilities (e.g., accessibility facilities), and connections.

The **IT2Rail** ontology was developed within the project with the same name. The ontology is at the core of the innovative framework developed within the project, which enables interoperability among existing transportation systems without prerequisites for centralized standardization.

**NeTEx** is a CEN technical standard for exchanging public transport schedules and related data. It is divided into three parts, each covering a functional subset of Transmodel. The standard covers many aspects within its scope, but we focus on information regarding stops, routes, timetables, fares, and schedules.

The **FSM** specification represents information about ticketing and reservations in a heterogeneous transport environment. More precisely, it provides a framework to support the distribution of rail passenger products, which allows companies to distribute their products via (or combined with the products of) other operators or ticket vendors. Our experiments focus on the parts of the specification that capture information related to itineraries, stops, vehicles, fares, and timings.

Finally, the **Trias** specification was introduced by the Verband Deutscher Verkehrsunternehmen (VDV) [41]. It covers a wide range of topics, such as positions of vehicles and passengers, disruptions, departure/arrival events, fares, connections, and journeys. In our case studies, we used a subset of the Trias standard, covering the topics of stations,

location search, departures, navigation, and ticket price calculation.

### B. VALIDATION METHODOLOGY

We designed six test cases by combining different pairs of data specifications (presented in Section V-C and summarized in Table 6). Each test case uses an XSD-based specification as the "source" specification and an ontology as the "target" one. This choice derives from the consideration that one of the main uses that we envisage for the technique developed in this paper is the creation of mappings towards a "pivot" ontology to be used in the frame of approaches such as the one presented in [7]. Hence, to evaluate the accuracy of our technique, we check its ability to suggest proper mappings in the target ontology starting from the terms in the source XSD specification. As a consequence, in the following, when referring to "the total set of terms,", we consider the terms appearing in the source specification—i.e., those for which we are actually looking for a mapping.

For each test case, we have carefully assessed the output results to determine the accuracy of the mapping suggestions. To this end, for each selected pair of specifications, we relied on the available documentation to create the "ground truth", that is, the expected mappings, between terms of the two specifications. We remark that in the creation of the expected mappings there is a degree of subjectivity that cannot be avoided, as different people could interpret the meaning of a term differently, even given the same documentation. To automatically check the mappings suggested by our tool against the ground truth we have prepared, for each pair of specifications, a file listing the expected mappings (and an explanation of the rationale for the identified mapping). For each pair of selected specifications, we have executed the tool for each version of the Transport model listed in Table 4 and also for the Google news model, and we collected the suggested mappings.

To analyze the results, we categorized each term TermS of the source specification as *Correct*, *Incorrect*, *Unfeasible*, or *Leftout*, depending on whether the tool was able to find a proper mapping ⟨TermS, TermT⟩ for it or not (as mentioned above, the ultimate goal is to find correspondences in the target ontology for terms appearing in the source XSD specification). More precisely, we label a term TermS as *Correct* when there is a mapping ⟨TermS, TermT⟩ that is presented as one of the top three suggestions presented by our tool (these are indeed the pairs actually shown to the user, as explained in [8]), and the pair also appears in the validation file capturing the ground truth. We label TermS as *Incorrect* when none of the mappings ⟨TermS, TermT⟩ suggested for it appears in the validation file, but the latter contains another—correct—mapping for ⟨TermS, TermT′⟩. A term TermS is *Unfeasible* if there is no pair ⟨TermS, TermT⟩ for it in the validation file. A term of the source specification that is not *Correct*, nor *Incorrect*, nor *Unfeasible* is tagged as *Leftout*. That is, a term is *Leftout* if no suggestion is ultimately provided for it because the suggested mapping is filtered out at some stage in the process (which includes the case in which one of the two terms of the correct mapping does not appear in the trained model). Notice that a *Leftout* term is one for which the mapping creation procedure is inconclusive since the mapping tool does not have enough information to venture a suggestion, but no incorrect mapping is suggested.

To calculate the accuracy of the model, we use two criteria termed *Accuracy* and $Accuracy_{total}$, respectively. The value of *Accuracy* is computed, as shown in Formula (2), taking into account only correct and incorrect pairs, as the share of correct pairs with respect to the total number of pairs for which a clear categorization (correct or incorrect) was given.

$$Accuracy(\%) = \frac{Correct}{Correct + Incorrect} * 100 \tag{2}$$

The formula (3) defines how the value of $Accuracy_{total}$ is calculated. As shown by the formula, in this case, also terms labeled *Leftout* are considered, to consider also situations in which the tool does not venture a conclusive answer.

$$Accuracy_{total}(\%) = \frac{Correct}{Correct + Incorrect + Leftout} * 100 \tag{3}$$

### C. VALIDATION RESULTS

The six pairs of specifications used to validate our approach are the following: (i) FSM to Transmodel; (ii) Trias to Transmodel; (iii) NeTEx to Transmodel; (iv) NeTEx to IT2Rail; (v) Trias to IT2Rail; (vi) FSM to IT2Rail. Table 6 summarizes the results of the experiments, which are described in the rest of this section, and which are then discussed in Section V-D (the raw results themselves are available at [35]).

#### 1) TestCase 1. FSM TO TRANSMODEL

Figure 6 shows the number of *Correct*, *Incorrect*, *Unfeasible* and *Leftout* terms obtained with the various models. As shown in the figure, there are 55 *Unfeasible* terms. Indeed, many of

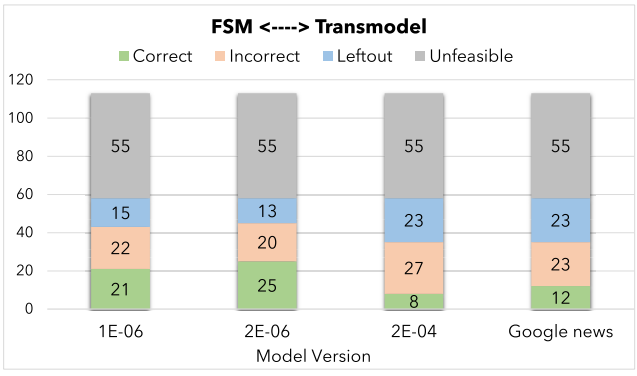


**FIGURE 6.** Case 1: Results of the FSM to transmodel mapping.

the concepts that appear in the FSM standard (for example, "Coach Id", "Compartment Id", "Mobility Aid Type Id") do not have a corresponding term in Transmodel. In fact,

the Transmodel ontology mostly contains information about routes, lines, journey patterns, and information related to stop points, etc. Table 6 shows the computed values for *Accuracy* and $Accuracy_{total}$.

In this case, Transport model version 2E-6 has the highest value, 55%, for *Accuracy*, whereas the second-best model is version 1E-6, which has an accuracy of 49%. On the other hand, the table shows that Google news model has only 34% accuracy.

### 2) TestCase 2. TRIAS TO TRANSMODEL

The results of this experiment are shown in Figure 7. Also in this case, there is a considerable number of *Unfeasible* terms. For example, in the Transmodel ontology, there are no corresponding concepts for the terms "Booking Deadline" and "Fare Zone Text", since the specification does not cover topics related to booking, whereas those related to fares are not covered with the same level of detail as in the Trias standard.

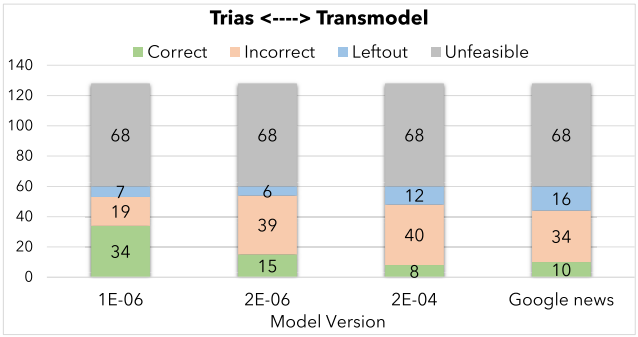


**FIGURE 7.** Case 2: Results of the trias to transmodel mapping.

Transport model version 1E-6 exhibits the highest accuracy, 64%, as shown in Table 6, while the second-best one is Transport model version 2E-6, for which the accuracy is 28%.

### 3) TestCase 3. NeTEx TO TRANSMODEL

Figure 8 shows the results of this test case for different models. As in all test cases, in addition to a number of *Unfeasible*, there are also a few *Leftout* terms. Indeed, some of the phrases from the NeTEx standard are filtered out by the tool at the very initial stage because one or more of the words appearing in the phrase does not belong to the vocabulary of the trained model (this is the case, for example, for phrases "Group Of Lines" and "Type Of Sales Package").

Given these statistics in this case, Transport model version 1E-6 model shows the highest (46%) accuracy, whereas Google news model exhibits 45% accuracy as shown in the Table 6.

### 4) TestCase 4. NeTEx TO IT2Rail

Figure 9 shows the results of the mapping process for this case study. The high number of *Unfeasible* terms is due to the fact that the NeTEx specification covers topics related to fares,

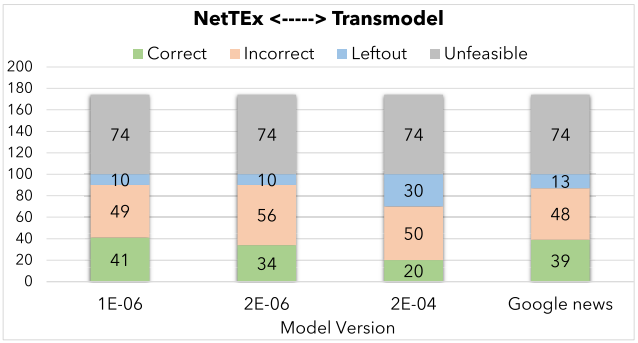


**FIGURE 8.** Case 3: Results of the NeTEx to transmodel mapping.

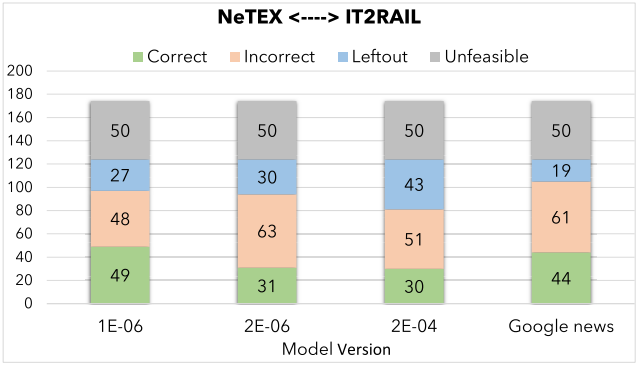


**FIGURE 9.** Case 4: Results of the NeTEx to IT2Rail mapping.

transport agencies and geographical information with a level of detail that is much higher than in the IT2Rail ontology; hence many source terms do not have a corresponding concept in the target standard. In addition, as shown in the figure, there are quite a few *Leftout* terms. Many of them were filtered out by the tool because one or more of the words appearing in the phrase do not belong to the vocabulary of the trained model (e.g, "logical displays").

Considering the feasible terms, Transport model's version 1E-6 provides the most accurate results (with a value of *Accuracy* equal to 50%, as shown in Table 6), while Google news is the second-best model, with a value of *Accuracy* of 42%.

### 5) TestCase 5. TRIAS TO IT2Rail

The results of this case study are shown in Figure 10. As can be seen in the figure, there are quite a few *Incorrect* terms. A manual inspection of the results showed that this phenomenon was, in particular, relevant for n-grams. For example, the term "Trip Leg Structure" (which refers to the structure of passenger trips) is mapped to "Air Transport", though the correct mapping should be "Itinerary". In this (and other) cases, the phrases to be mapped to one another are very different, and one can imagine that, in the texts used for training the models, they appear in contexts that are related but not very similar. Hence, the mapping tool ultimately

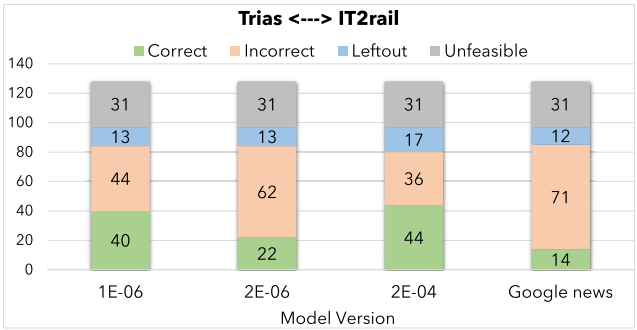


**FIGURE 10.** Case 5: Results of the Trias to IT2Rail mapping.

provides a suggestion, but it is not accurate because of the context difference.

A comparison of the results obtained with the different models shows that Transport model version 2E-4 has the highest accuracy (55%), while Transport model version 1E-6 provides the second-best accuracy (48%).

#### 6) TestCase 6. FSM TO IT2Rail

The results of the mapping process for this case study are shown in Figure 11. The figure highlights that Transport model version *1E-6* is, by far, the best model for this case. In particular, out of 82 feasible terms, model version 1E-6 accurately maps 52 of them, for a value of *Accuracy* of 68% (see Table 6); Transport model version 2E-6 is the second-best in terms of performance, with *Accuracy* equal to 41%.

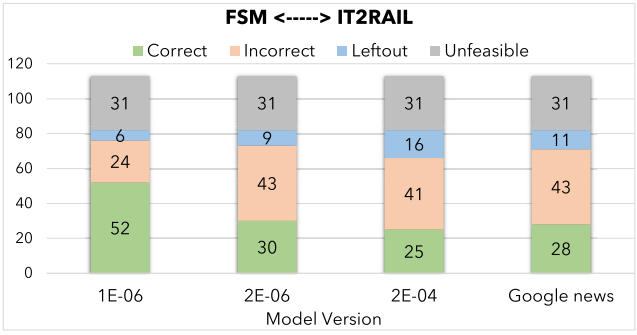


**FIGURE 11.** Case 6: Results of the FSM to IT2Rail mapping.

### D. DISCUSSION

The results of the test cases presented above, which are summarized in Table 6, show that, with respect to research question Q1, the mapping tool is indeed able to provide reasonably accurate suggestions when it is able to create a suggestion with sufficient confidence.

In particular, in all test cases, there is at least one version of the domain-specific Transport model that shows a value of *Accuracy* that is greater than 45% (indeed, in most cases, it is higher than 50%). Even if one considers the value of $Accuracy_{total}$, which takes into account also *Leftout* terms, for which the tool is not able to provide a suggestion with enough

**TABLE 6.** Accuracy comparison between different model versions (where Act = $Accuracy_{total}$ and Ac = *Accuracy*). Dark grey cells correspond to the W2V-trained model that shows the highest level of accuracy, whereas light grey cells highlight the second-best one.

| Testcases | 1E-6 | | 2E-6 | | 2E-4 | | GN | |
|---|---|---|---|---|---|---|---|---|
| | Ac | Act | Ac | Act | Ac | Act | Ac | Act |
| *FSM - Transmodel* | 49% | 36% | 55% | 43% | 23% | 14% | 34% | 20% |
| *Trias - Transmodel* | 64% | 56% | 28% | 25% | 16% | 13% | 23% | 16% |
| *NeTEx - Transmodel* | 46% | 41% | 38% | 34% | 28% | 20% | 45% | 39% |
| *NeTEx - IT2Rail* | 50% | 39% | 33% | 25% | 37% | 24% | 42% | 35% |
| *Trias - IT2Rail* | 48% | 41% | 26% | 23% | 55% | 45% | 16% | 14% |
| *FSM - IT2Rail* | 68% | 63% | 41% | 36% | 38% | 30% | 39% | 34% |

confidence, the accuracy is higher than 45% in half the cases and close to that value in two other cases.

A possible way to reduce the number of *Leftout* terms and thus increase the value of $Accuracy_{total}$ is to expand the range of phrases included in the W2V-trained model at the core of the mapping mechanism. This could be achieved by providing a larger dataset to the training algorithm, thus covering a wider range of concepts. Nevertheless, even with a fairly small (around 12K phrases) but domain-focused model, the level of accuracy (both *Accuracy* and $Accuracy_{total}$) can be deemed satisfactory.

Concerning research question Q2, Table 6 highlights that, in each case, the general-purpose Google News model is outperformed by at least one version of the domain-specific Transport model (the table highlights the best-performing model through dark grey cells and the second-best model with light grey cells). In particular, the table shows that Transport model version 1E-6 has the highest accuracy in four out of six test cases; versions 2E-6 and 2E-4, instead, are most accurate in one test case each (FSM-Transmodel and Trias-IT2Rail, respectively), though version 1E-6 is, in any case, the second best. Overall, then, we can conclude that Transport model 1E-6 has the best accuracy among all models, which shows the benefits of using a domain-specific model with respect to a general-purpose one.

Given the experiments carried out, we can identify several **threats to the validity** of the results. The first threat lies in the fact that we ran a limited number of test cases, focusing on a single domain (transportation). However, the transportation domain is very complex and heterogeneous, with a wide range of standards used by different actors. As such, it can be considered a paradigmatic example of a complex environment, and we are confident that the issues and difficulties that it highlights are highly representative also

for other domains. In this context, we chose relevant standards used in the transportation domain, and we considered all pairings of one XSD-based specification and one ontology, so, in this sense, the tests carried out were exhaustive. A second threat to the validity of the obtained results lies in the correctness of the validation file that we used as ''ground truth'' to evaluate the accuracy of the mapping mechanism. In this regard, we first notice that, as also mentioned in Section V-B, there is a level of subjectivity in the interpretation of the standards, and different, reasonable people could come up with different mappings if they were created by hand; hence, an ''ultimately correct mapping'' does not really exist, but one can say that there are ''several reasonable ones''. In particular, to create the ground truth, we relied on our own understanding of the standards, and we made our best effort to identify a plausible mapping between the concepts. In addition, in the validation workflow, we first created the validation file, and only after that, we ran the experiments with the mapping tool. This should reduce considerably, if not eliminate entirely, any bias that there might be in the creation of the ground truth and avoid the possibility that the validation file were skewed to artificially increase the accuracy of our tool.

## VI. CONCLUSION

This article presented in detail and extended the mechanisms for the automated generation of mappings between concepts in different data specifications that underlie the SMART tool. Our mapping technique is unique in that it follows a two-fold approach by combining linguistic similarity while also taking into account the structural position of concepts in specifications. In this paper, we also showed how the training of a custom, domain-specific model for word embeddings could help increase the accuracy of the generated mappings. The work presented in this article can help increase the level of interoperability between heterogeneous data specifications used by different actors in a specific domain.

In future work, we plan to improve the linguistic and structural mechanisms underpinning our approach. More precisely, we will improve the linguistic part, on the one hand, by training our models on bigger corpora and, on the other hand, by exploring the effectiveness of alternative word embedding techniques. In addition, we will enhance the structural part of the technique by covering more aspects of the structure of the data specifications. Finally, we plan to apply the proposed technique to domains other than the transportation one.

## REFERENCES

[1] M. Jamshidi, *Systems of Systems Engineering: Principles and Applications*. Boca Raton, FL, USA: CRC Press, 2017.
[2] P. Uday and K. Marais, ''Designing resilient systems-of-systems: A survey of metrics, methods, and challenges,'' *Syst. Eng.*, vol. 18, no. 5, pp. 491–510, Oct. 2015, doi: 10.1002/sys.21325.
[3] M. Sadeghi, A. Carenini, O. Corcho, M. Rossi, R. Santoro, and A. Vogelsang, ''Interoperability of heterogeneous systems of systems: Review of challenges, emerging requirements and options,'' in *Proc. Symp. Appl. Comput. (SAC)*, 2023, doi: 10.1145/3555776.3577692.
[4] P. Jittrapirom, V. Caiati, A.-M. Feneri, S. Ebrahimigharehbaghi, M. J. A. González, and J. Narayan, ''Mobility as a service: A critical review of definitions, assessments of schemes, and key challenges,'' *Urban Planning*, vol. 2, no. 2, pp. 13–25, Jun. 2017, doi: 10.17645/up.v2i2.931.
[5] *Roadmap to a Single European Transport Area Towards a Competitive and Resource Efficient Transport System*, European Commission, White Paper. Accessed: Mar. 24, 2023. [Online]. Available: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2011:0144:FIN:EN:PDF
[6] *Shift2Rail*. Accessed: Nov. 20, 2022. [Online]. Available: https://projects.shift2rail.org/
[7] M. Scrocca, M. Comerio, A. Carenini, and I. Celino, ''Turning transport data to comply with EU standards while enabling a multimodal transport knowledge graph,'' in *Proc. ISWC 19th Int. Semantic Web Conf., II*, J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, and L. Kagal, Eds. Athens, Greece, 2020, pp. 411–429, doi: 10.1007/978-3-030-62466-8_26.
[8] S. Kalwar, ''SMART: Towards automated mapping between data specifications,'' in *Proc. 33rd Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2021, pp. 429–436, doi: 10.18293/seke2021-161.
[9] S. Pereira Detro, D. Morozov, M. Lezoche, H. Panetto, E. P. Santos, and M. Zdravkovic, ''Enhancing semantic interoperability in healthcare using semantic process mining,'' in *Proc. 6th Int. Conf. Inf. Soc. Techol., (ICIST)*, vol. 1. Kopaonik, Serbia, Feb. 2016, pp. 80–85.
[10] O. Iroju, A. Soriyan, I. Gambo, and J. Olaleke, ''Interoperability in healthcare: Benefits, challenges and resolutions,'' *Int. J. Innov. Appl. Stud.*, vol. 3, no. 1, pp. 262–270, 2013.
[11] M. Ehrig, A. Koschmider, and A. Oberweis, ''Measuring similarity between semantic business process models,'' in *Proc. 4th Asia–Pacific Conf. Comceptual Modelling*, vol. 67, 2007, pp. 71–80.
[12] S. K. Lau, R. Zamani, and W. Susilo, ''A semantic web vision for an intelligent community transport service brokering system,'' in *Proc. IEEE Int. Conf. Intell. Transp. Eng. (ICITE)*, Aug. 2016, pp. 172–175, doi: 10.1109/ICITE.2016.7581328.
[13] D. Chaves-Fraga, P. Colpaert, M. Sadeghi, and M. Comerio, ''Editorial of transport data on the web,'' *Semantic Web*, pp. 1–4, Oct. 2022, doi: 10.3233/SW-223278.
[14] N. K. Y. Leung, S. K. Lau, and J. Fan, ''Enhancing the reusability of inter-organizational knowledge: An ontology-based collaborative knowledge management network,'' in *Proc. 5th Int. Conf. Intellectual Capital, Knowl. Manag. Organisational Learn., (ICICKM)*. New York, NY, USA: Academic, 2008, p. 269, doi: 10.58729/1941-6687.1094.
[15] W.-D. Yang and T. Wang, ''The fusion model of intelligent transportation systems based on the urban traffic ontology,'' *Phys. Proc.*, vol. 25, pp. 917–923, Jan. 2012, doi: 10.1016/j.phpro.2012.03.178.
[16] N. K. Y. Leung, S. K. Lau, and N. Tsang, ''An ontology-based collaborative inter-organisational knowledge management network (CIK-NET),'' *J. Inf. Knowl. Manage.*, vol. 12, no. 1, Mar. 2013, Art. no. 1350005, doi: 10.1142/S0219649213500056.
[17] T. Sobral, T. Galvão, and J. Borges, ''Semantic integration of urban mobility data for supporting visualization,'' *Transp. Res. Proc.*, vol. 24, pp. 180–188, Jan. 2017, doi: 10.1016/j.trpro.2017.05.106.
[18] N. O. Pinciroli Vago, M. Sacaj, M. Sadeghi, S. Kalwar, A. Vogelsang, and M. G. Rossi, ''On the visualization of semantic-based mappings,'' in *Proc. 3rd Int. Workshop Semantics Web Transp., Co-Located With Semantics Conf. (SEMANTiCS)*, vol. 2939, D. Chaves-Fraga, P. Colpaert, M. Sadeghi, M. Scrocca, and M. Comerio, Eds. Sep. 2021, pp. 1–10.
[19] T. Rodrigues, P. Rosa, and J. S. Cardoso, ''Mapping XML to existing OWL ontologies,'' in *Proc. IADIS Int. Conf. WWW/Internet*, 2006, pp. 72–77.
[20] M. Ferdinand, C. Zirpins, and D. Trastour, ''Lifting XML schema to OWL,'' in *Proc. Web Eng., 4th Int. Conf., ICWE*, vol. 3140, N. Koch, P. Fraternali, and M. Wirsing, Eds. Munich, Germany: Springer, Jul. 2004, pp. 354–358, doi: 10.1007/978-3-540-27834-4_44.
[21] R. García, F. Perdrix, and R. Gil, ''Ontological infrastructure for a semantic newspaper,'' in *Proc. Semantic Web Annotations Multimedia Workshop (SWAMM)*, 2006, pp. 1–12.
[22] J. Pennington, R. Socher, and C. D. Manning, ''GloVe: Global vectors for word representation,'' in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar, Oct. 2014, pp. 1532–1543, doi: 10.3115/v1/d14-1162.
[23] M. E. Peters, M. Neumann, M. Iyyer, and M. Gardner, ''Deep contextualized word representations,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1. New Orleans, LA, USA: ACL, Jun. 2018, pp. 2227–2237, doi: 10.18653/v1/n18-1202.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186, doi: 10.18653/v1/n19-1423.
[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[26] *Word2Vec_Implementation*. Accessed: Mar. 24, 2023. [Online]. Available: github.com/rahul1728jha/Word2Vec_Implementation
[27] *Gensim Library*. Accessed: Mar. 24, 2023. [Online]. Available: https://radimrehurek.com/gensim/
[28] *Word2Vec Tool*. Accessed: Mar. 24, 2023. [Online]. Available: https://code.google.com/archive/p/word2vec/
[29] *XML Schema Definition Language (XSD)*. Accessed: Mar. 24, 2023. [Online]. Available: https://www.w3.org/TR/xmlschema11-1/
[30] *Web Ontology Language (OWL)*. Accessed: Mar. 24, 2023. [Online]. Available: https://www.w3.org/OWL
[31] *Intelligent Transport*. Accessed: Mar. 24, 2023. [Online]. Available: https://www.intelligenttransport.com/magazine/
[32] *Metro*. Accessed: Mar. 24, 2023. [Online]. Available: https://www.metro-magazine.com
[33] E. S. Prassas and R. P. Roess, *Engineering Economics and Finance for Transportation Infrastructure* (Springer Tracts on Transportation and Traffic). Cham, Switzerland: Springer, 2013, doi: 10.1007/978-3-642-38580-3.
[34] V. Profillidis, *Railway Management and Engineering*. Evanston, IL, USA: Routledge, 2016.
[35] *SMART Data Repository*. Accessed: Mar. 24, 2023. [Online]. Available: https://github.com/safia-k/SprintData
[36] *NeTEx (Network Timetable EXchange) XML Schema*. Accessed: Mar. 24, 2023. [Online]. Available: https://github.com/NeTEx-CEN/NeTEx
[37] *FSM Full Service Model*. Accessed: Mar. 24, 2023. [Online]. Available: https://tsga.eu/fsm
[38] *TRIAS*. Accessed: Mar. 24, 2023. [Online]. Available: https://opentransportdata.swiss/en/cookbook/triprequest/
[39] *Transmodel Ontology Organisations*. Accessed: Mar. 24, 2023. [Online]. Available: https://oeg-upm.github.io/snap-docs/tm-organisations.owl/documentation/index-en.html
[40] *CEN European Reference Data Model for Public Transport Information*. Accessed: Mar. 24, 2023. [Online]. Available: https://www.transmodel-cen.eu/
[41] *Vdv*. Accessed: Mar. 24, 2023. [Online]. Available: https://www.vdv.de/

**SAFIA KALWAR** is currently pursuing the Ph.D. degree in computer science with Politecnico di Milano. Her research interests include semantic web, machine learning, and security of big data.

**MATTEO ROSSI** is currently an Associate Professor with Politecnico di Milano. His research interests include formal methods for safety-critical and real-time systems, architectures for real-time distributed systems, and transportation systems both from the point of view of their design and their application in urban mobility scenarios.

**MERSEDEH SADEGHI** received the Ph.D. degree in computer science from Politecnico di Milano. She is currently a Postdoctoral Researcher with the University of Cologne. Her research interests include software engineering, distributed systems, semantic-aware systems, and the explainability of context-aware pervasive systems.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement