Requirements Engineering
https://doi.org/10.1007/s00766-021-00366-0

ORIGINAL ARTICLE

A multi-level semantic web for hard-to-specify domain concept, *Pedestrian*, in ML-based software

Hamed Barzamini[1] · Murtuza Shahzad[1] · Hamed Alhoori[1] · Mona Rahimi[1]

Received: 11 November 2020 / Accepted: 30 October 2021

Abstract
Machine Learning (ML) algorithms are widely used in building software-intensive systems, including safety-critical ones. Unlike traditional software components, Machine-Learned Components (MLC)s, software components built using ML algorithms, learn their specifications through generalizing the common features that they find in a limited set of collected examples. While this inductive nature overcomes the limitations of programming *hard-to-specify* concepts, the same feature becomes problematic for verifying safety in ML-based software systems. One reason is that, due to MLCs data-driven nature, there is often no set of explicitly written and pre-defined specifications, against which the MLC can be verified. In this regard, we propose to partially specify hard-to-specify domain concepts, which MLCs tend to classify, instead of fully relying on their inductive learning ability from arbitrarily-collected datasets. In this paper, we propose a semi-automated approach to construct a multi-level semantic web to partially outline the hard-to-specify, yet crucial, domain concept "pedestrian" in automotive domain. We evaluate the applicability of the generated semantic web in two ways: first, with a reference to the web, we augment a pedestrian dataset for a missing feature, *wheelchair*, to show training a state-of-the-art ML-based object detector on the augmented dataset improves its accuracy in detecting pedestrians; second, we evaluate the coverage of the generated semantic web based on multiple state-of-the-art pedestrian and human datasets.

Keywords Requirements specifications · Machine learning · Machine-learned components · Safety-critical systems

1 Introduction

Software components are traditionally built according to a set of predefined requirements specifications. These specifications are expected to explicitly and unambiguously describe what a software component is expected to do in a given context. Requirements are usually gathered based on stakeholders' and users' needs during the domain analysis and requirements engineering phases. However, when engineering domain-specific software systems, requirements are often expressed containing domain-specific terminologies and concepts that are present in a particular environment in which the software operates. In conventional Requirements Engineering (RE), if needed, these domain-specific concepts and terminologies are typically specified during domain analysis and documented, alongside requirements, either in the form of natural language or in a more formal manner, such as an ontology, taxonomy, and semantic webs [11, 12, 17, 30, 49].

With the emergence of Machine Learning (ML) applications in Software Engineering (SE), domain concepts are no longer explicitly specified. The concepts are instead implicitly represented within limited variations of concept's instances in a training dataset. While conventional requirements engineers apply elicitation techniques to obtain and validate domain knowledge, the use of ML in SE has turned this process to be more of an inductive and data driven task. As such, ML-based software components, Machine-Learned Components (MLC)s, learn their specifications from a set of collected examples rather than a set of "agreed upon" specifications. For instance, pedestrian

✉ Mona Rahimi
rahimi@cs.niu.edu

Hamed Barzamini
hbarzamini@niu.edu

Murtuza Shahzad
msyed1@niu.edu

Hamed Alhoori
alhoori@cs.niu.edu

[1] Northern Illinois University, DeKalb, IL 60115, USA

detectors in autonomous technology, learn the concept *pedestrian* solely from images and video frames of various pedestrians in a training set; in medical domain, a ML-based diagnostic software tends to learn the specifications of a benign versus a malignant tumor only from a collection of Computed Tomography(CT) scans. Hence, ML models suggest specifications (inductively learned from the training-validation dataset) rather than implementing known specifications. This characteristic makes MLCs particularly useful when a domain consists of concepts, where little knowledge exists for humans to develop effective algorithms.

We refer to such domain-specific concepts as *hard-to-specify* for which a universal definition does not exist, as they are inherently challenging to clearly define. Therefore, these are domain concepts with no clearly delineated and agreed-upon definition due to the large difference that exists between varying instances of the concept. Although humans may recognize various instances of the concept through *intuition*, yet the concepts' definite specification is difficult even for humans. For instance, what is the exact specification for recognizing a potential pedestrian? In a potential requirement, "the determined position of the '*pedestrian*' should be accurate within 0.5 m," [45] but the term '*pedestrian*' is not simply determinable. The concept '*pedestrian*' is hard to specify due to pedestrians' different appearances in the visual domain caused by clothes and/or additional equipment.

With the advent of autonomous cars, MLCs are used in software systems within autonomous driving technology, such as in the Advanced Driver Assistance Systems (ADAS) and the Automated Driving Systems (ADS) [27]. In such a context, ML-based systems become safety-related as their failure may result in loss of life or severe damage to properties and/or environment [23]. MLCs' inductive ability to learn specifications is desirable for programming hard-to-specify concepts in this domain. However, the same ability may also result in a significant and yet inevitable gap between the perception of concepts (e.g., '*pedestrian*') as human drivers perceive and what a collected dataset represents as the concept for MLCs' training process. MLCs' perception of concepts is confined to generalizing a set of seen common features within a limited dataset, collected for training purposes of the model. For instance, a pedestrian detector in a smart car is partially expected to learn the concept '*pedestrian*' based on images and video streams in which pedestrians are annotated. If the real pedestrians in the model's operation environment do not share the same learned common features of pedestrians that the model has seen during the development process, the detector misclassifies the pedestrians as non-pedestrians. Thus, using MLCs in this domain and relying on their learning ability from a limited dataset, which may not cover all factors related to the concept '*pedestrian*', introduces safety risks.

In the automotive industry, the applicable functional safety standard is referred to as ISO 26262 [20], which does not cover specific aspects of ML [37]. This was a key motivation for the extension ISO/PAS 21448 on Safety Of The Intended Functionality (SOTIF). However, there is no clear guideline in the document on precisely addressing the mentioned issues in MLCs [21]. The state-of-the-art pedestrian datasets are collected in unsystematic and based on ad hoc manners and therefore are generally limited in the number of examples and diversity of samples that they comprise [21, 46]. For instance, the most recently established datasets in the context of autonomous driving, such as Caltech [14], KITTI [18], CityPersons [53], and EuroCityPerson (ECP) [6], are arbitrarily collected by a vehicle-mounted camera navigating through suburban roads [21].

In this regard, in this paper, we propose to partially specify hard-to-specify domain concept, *pedestrian*, for MLCs instead of entirely relying on their ability to learn these specifications from a randomly-collected dataset. Specifying domain concepts improve requirements engineering tasks in developing MLCs by (i) enriching their learning foundation with semantic information (ii) providing a more complete basis to learn functional requirements specifications (iii) disambiguating MLC's requirements specifications which contain hard-to-specify domain terms. To this end, we introduce a novel semi-automated approach to construct a multi-level semantic web for partially specifying a potential pedestrian for MLCs. A process-level overview of our proposed approach is illustrated in Fig. 1, containing seven phases. Figure 2 uses the same corresponding phase numbers to represent a more detailed development-level overview of each step. As shown in both figures, our approach contains two major parts **Designing the Semantic Web** and **Augmenting the Dataset**. In part "Designing the Semantic Web", we specify the domain concept, pedestrian, first by augmenting the concept through searching for accompanying, co-occurring, and related similar terms, Fig. 1-(1). Second, we create a set of high-level topics (Zoom-Out view) from the retrieved terms by following a sequence of ML techniques, Fig. 1-(2). Third, more details are provided for each topic (Zoom-In view), using a part-of-speech tagger, Fig. 1-(3). Finally, instead of merely returning a flat list of relevant terms and relationships, we compose the terms into a meaningful hierarchy of topics in the form of a *semantic web*, Fig. 1-(4). In part "Augmenting the Dataset", we assess the *usefulness* of our approach through showing the *effectiveness* of the inferred semantic web in improving MLCs' accuracy, as well as empirically analyzing the *coverage* of the generated topics and relationships in benchmark datasests. First with a reference to the web, we identified one semantically important feature, wheelchair, for pedestrians that appeared in the semantic web but was not present in the majority of commonly-used datasets, Fig. 1-(5). We then manually

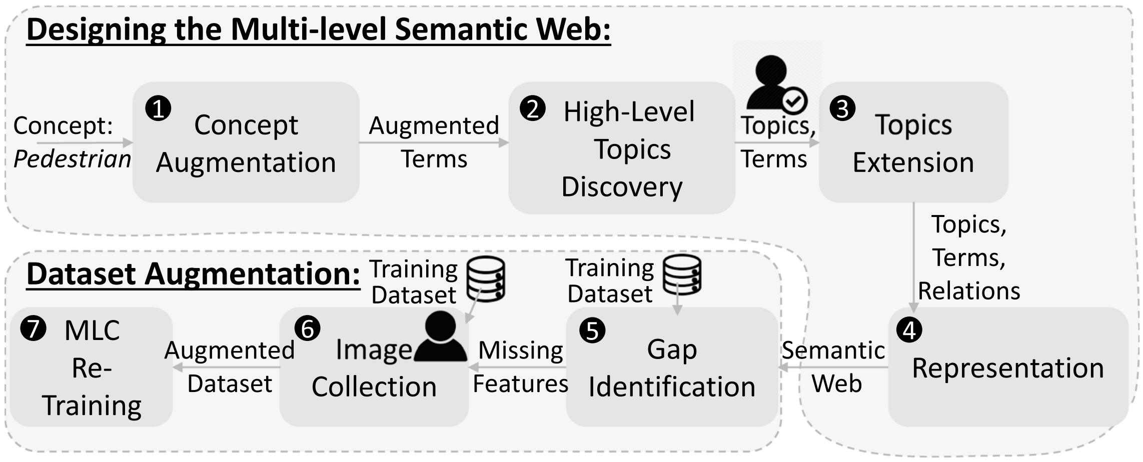

Fig. 1 Process-level overview of our proposed approach

collected a balanced number of images of pedestrians in wheelchair and augmented the MLC's original training dataset, Fig. 1-(6). Finally, we re-trained a state-of-the-art object detector to detect the augmented feature, and evaluated the **effectiveness** of our approach in improving MLCs learning process, through conducting a series of experiments before and after the dataset augmentation, Fig. 1-(7). Additionally, to assess the **coverage** of the semantic web that was generated by the proposed approach, we analyzed multiple state-of-the-art datasets with respect to entities and relationships built in the semantic web.

The primary contribution of this work is, therefore, a paradigm shift to formally specify hard-to-specify domain concepts and partial requirements for MLCs. The advantages of outlining hard-to-specify domain concepts in the form of a semantic web are (i) to set a benchmark for the specification of unambiguous hard-to-specify concepts, (ii) to build a library to reuse and augment the shared knowledge, (iii) to automatically verify whether the datasets and ML models are consistent with the specifications, given that the formalism form of the knowledge is represented in a semantic web. Thus, we can use the established benchmarks to serve as a reference for assessing the correctness and completeness of a concept captured in a collected dataset for training-validating-testing MLC, (iv) to improve the process of specifying unambiguous requirements for MLCs. For example, in the requirement, "The determined position of the '*pedestrian*' should be accurate within 0.5 m", outlining the term '*pedestrian*' relatively resolves the inherited ambiguity of domain-related concept '*pedestrian*' and results in a more explicit requirement specification.

In this paper, we selected the automotive domain and pedestrian as a hard-to-specify domain concept mainly because most safety-related perceptional requirements for an intelligent pedestrian detector are based on the incorrect assumption that such domain specification exists and is agreed upon. We use our proposed approach to semi-automatically build a semantic web for the hard-to-specify concept '*pedestrian*'. We then show that augmenting a dataset, according to our established semantic web, improves the accuracy of a pedestrian classifier trained on this dataset.

The remainder of this paper is organized in the following manner. In Sect. 2, we discuss our semi-automated process to design a multi-level semantic web fro specifying hard-to-specify concepts. We describe how we created a high-level view (Zoom-Out view), and the process of creating a more detailed view (Zoom-In View) of the concept '*pedestrian*'. We finish this section by representing the process of creating a semantic web from the words and relations previously retrieved. Section 3 describes how we applied our inferred semantic web to augment a publicly available pedestrian dataset in order to improve the performance of selected pedestrian detectors. Sections 4, 5, and 6 discuss threats to the validity of our approach, related work, and conclusion and future work, respectively.

2 Designing the semantic web

(Figure 1-(1)) In this section, we present our approach to design a semantic web with the purpose of formally specifying hard-to-specify domain concepts. While we use our approach to semi-automatically construct an extendable domain-specific semantic web for the concept, '*pedestrian*' in the automotive domain, the process is not domain-specific and can be generalized to other concepts and domains. The output of this section, the constructed semantic web, serves as a benchmark for the concept '*pedestrian*' and a point of reference, against which collected '*pedestrian*' datasets for training ML algorithms can be assessed for completeness.

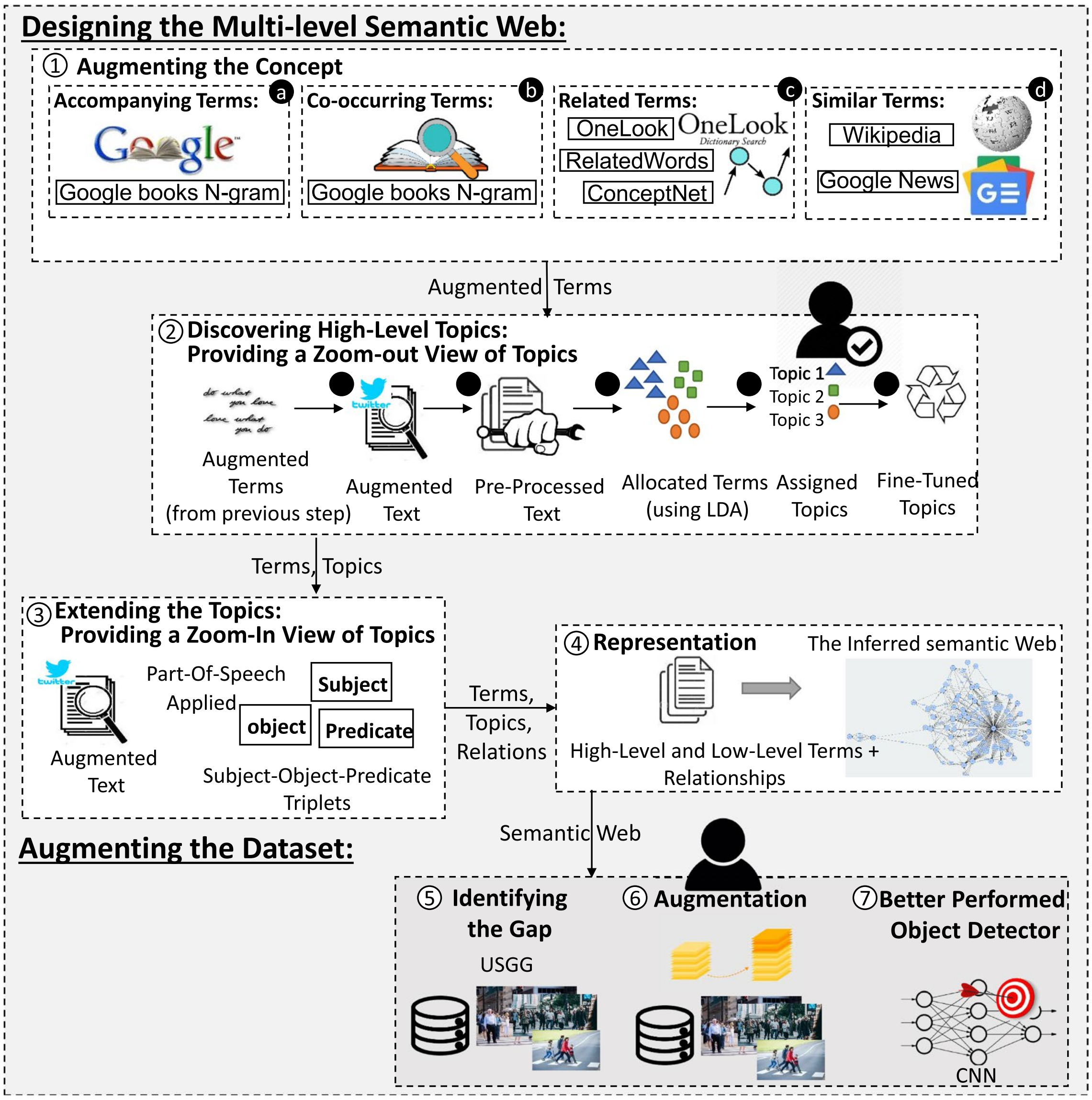

Fig. 2 Development-level overview of our proposed approach

2.1 Augmenting a concept

The process is initiated with the augmentation of the hard-to-specify domain concept '*pedestrian*'. A class of hard-to-specify concepts refers to terms in a domain that are more socially-constructed rather than being defined according to a definite, explicit, and written description. Moreover, the definition of pedestrian is highly subjective and, thus, differs from person to person depending on their perspective, past experience, and knowledge. For instance, is a person riding a bike; carrying a bike; or standing on a bike; a pedestrian? Therefore, to specify such concepts, for which no set of domain documents exists, we decided to refer to human perception as a reference model. For this purpose, we referred to platforms where humans easily share their views, thoughts, and knowledge, such as social platforms, news feeds, and online encyclopedias. Through mining such resources, we identified terms that are connected to the term pedestrian. To identify terms which are closely related to the given concept, we adopted three sets of approaches to achieve more enhanced and thorough results:

1. **Accompanying terms (Fig. 2(1)-(a)):** We defined accompanying terms as terms which most frequently appeared before and after the term pedestrian in high ranked online corpora. To identify such terms, we referred to Google books N-gram[1], an online search

[1] https://www.english-corpora.org/googlebooks/.

Table 1 Top ten words returned by different search queries on Google books N-gram

"Pedestrian"+ [any verb]	"Pedestrian" + [any noun]
Pedestrian crossing	Pedestrian traffic
Pedestrian walks	Pedestrian mall
Pedestrian killed	Pedestrian bridge
Pedestrian pass	Pedestrian street
Pedestrian moving	Pedestrian zone
[Any verb] + "pedestrian"	**[Any noun] + "pedestrian"**
Protect pedestrian	Child pedestrian
Warn pedestrian	Adult pedestrian
Hurrying pedestrian	Level pedestrian
Involving pedestrian	Street pedestrian
Encourage pedestrian	Block pedestrian

engine that provides a search for 155 billion words from American English and 34 billion words from British English and provides high-frequency terms associated with a given term as a search query. We searched Google books N-gram for sources printed between the years 1500 and 2000 in the American English language while using the word pedestrian as a part of our search query. We used four variations of search queries including: pedestrian + [any verb]; [any verb] + pedestrian; pedestrian + [any noun]; [any noun] + pedestrian. Table 1 shows some examples of the returned terms by the search engine for the variations mentioned. This step resulted in 692 non-unique terms accompanying with the term pedestrian.

2. **Co-occurring terms (Fig. 2(1)-(b)):** Relying solely on the terms extracted in the previous step, we potentially miss the terms which do not appear immediately before or after a given term but are still closely related to the concept. For instance, in the phrase "In cities with poor infrastructures, pedestrians who use *wheelchairs* often are forced to use the street." The terms *pedestrian* and *wheelchair* are potentially relevant (i.e., a pedestrian on a wheelchair is a pedestrian) but are not subsequent. To avoid such errors, we additionally defined and searched for co-occurring terms with the term pedestrian within the same phrase. We defined co-occurring terms as the most common terms, which occurred up to four terms apart from the term pedestrian in corpora. We assumed further apart terms have lower relevance to our search query and, therefore, less important to our semantic web. To achieve co-occurring terms, we searched Google N-gram to extract the most common verbs and nouns that appeared up to four terms apart from the term pedestrian, and this step resulted in 1,601 co-occurring terms. As such, we assumed that co-occurring terms in further distance than four spaces are mostly irrelevant to the concept and are, therefore, excluded from our study. We later compensate for further-apart-yet-relevant terms that were mistakenly removed by this assumption through adding the following metrics.
3. **Related terms (Fig. 2(1)-(c)):** We defined related terms as terms that have a meaningful relationship with the concept pedestrian but do not occur within a short distance or together with the term pedestrian in corpora. To search for such terms, we used RelatedWords[2], which is an open-source project. RelatedWords runs several algorithms, such as word embedding, to convert words into multidimensional real-valued vectors (often tens or hundreds of dimensions), representing their meanings. The generated vectors of the words are then compared to a database of pre-computed vectors according to a set of existing corpora. Each word's vector values are learned based on the word's usage in the given corpora and is placed in the space for these values. This allows words used in similar ways to have similar representations and closer distance in the space, naturally capturing their meaning. An additional algorithm used by RelatedWords crawls through ConceptNet [42] to retrieve additional associated terms. ConceptNet is a knowledge graph that collects its knowledge from many sources, such as Wiktionary, Open Mind Common Sense (OMCS) [41], and WordNet [5]. RelatedWords provided 453 words related to the term pedestrian. In addition to RelatedWords, we used Onelook[3], which indexes over a thousand online dictionaries and encyclopedias to return the words related to a search query. In addition to dictionaries and encyclopedia, Onelook internally works on Datamuse API to search various data sources, such as the CMU

[2] https://relatedwords.org/.

[3] https://www.onelook.com/.

pronouncing dictionary as a source of phonetic transcriptions; Corpus-based data to build a language model that scores candidate words by context; and WordNet for semantic lexical relations. Using pedestrian as the search query, Onelook returned 527 words related to the term. Although RelatedWords and Onelook use a few common data sources, such as WordNet, they returned a different set of terms for the search query pedestrian. Onelook returned 140 words in common with RelatedWords.

Merging Results: We merged the acquired (1) accompanying, (2) co-occurring, and (3) related terms to form 3273 words that have meaningful relationships with the term pedestrian. The process of removing duplicate terms led to 1207 unique words.

Computing Similarity Score (Fig. 2(1)-(d)): Once words were merged, and duplicates were removed, we used the Gensim library to compute a similarity score for each term with the term pedestrian. The goal of computing similarity scores was to filter the incorrectly retrieved terms and have a ranking value to prioritize and distinguish more relevant terms from less relevant ones. To compute similarity scores, we used the Gensim library, which includes various pre-trained mathematical models and a collection of public corpora, often used for unsupervised topic modeling and natural language processing. The library contains an implementation of the Word2vec models, a group of related models used to produce word embedding. Word2vec takes, as its input, a large corpus of text and produces a vector space, typically of several hundred dimensions. Each unique word in the corpus is being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space [33].

We used two public and widely used corpora, namely Wikipedia and Google News since they are built on a substantial number of words, 400,000 and 3 billion words, respectively. In the Gensim library, the terms included in these corpora are embedded in the form of vectors. The library provides a Word2vec two-layer neural network model which produces a set of similar terms to a given query, according to their cosine similarity score between -1 and 1. Table 2 shows the top 10 ($N = 10$) most similar terms to the term pedestrian from our corpora, Wikipedia, and Google News. We can observe that similar words differ between the two corpora. For example, the Word2vec model using Wikipedia returns words with a slightly higher similarity score than Google News. Out of the unique merged 1207 terms, only 907 terms were present in the Word2vec models constructed from Wikipedia, and 902 were present in Google News corpora. Therefore, we continued our process only with these 907 and 902 relevant terms with similarity scores retrieved from the two corpora. We noticed that the rest of the terms were among the least relevant terms to the concept '*pedestrian*'.

Table 2 Top ten similar words to pedestrian from Wikipedia and Google News corpora

Wiki terms	Similarity	Google terms	Similarity
Walkway	0.6928	Bicyclist	0.6166
Lanes	0.6808	Crosswalk	0.5942
Sidewalks	0.6572	Motorist	0.5460
Roadway	0.6411	Bike lanes	0.5416
Vehicular	0.6380	Pedestrian walkways	0.5328
Thoroughfare	0.6337	Bicycle lanes	0.5256
Subway	0.6296	Bikeway	0.5248
Underpass	0.6193	Traffic calming	0.5239
Overpass	0.6157	Roadway	0.5181
Parking	0.6129	Traffic	0.5173

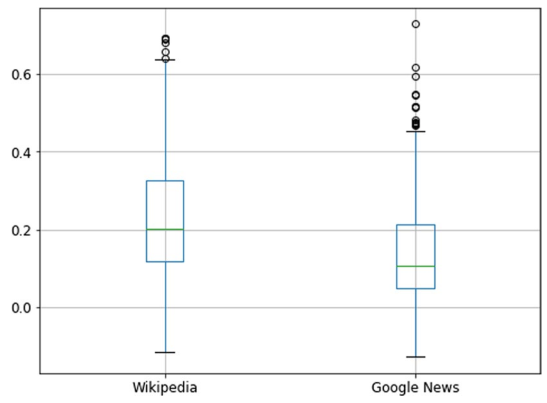

Fig. 3 Box plot of similarity scores for terms retrieved from Wikipedia and Google News

Filtering Results: We selected words with similarity scores within the upper quartile of all terms' similarity score boxplot to further eliminate the ineffectual terms from the remaining terms. Figure 3 represents the boxplots for both corpora, Wikipedia and Google News similarity scores. We found that the similarity score cutoff values (third quartile value) for Wikipedia and Google News were 0.3258 and 0.2137, respectively. After removing terms with similarity scores below the threshold limit, 227 terms from Wikipedia and 226 terms from Google News were obtained. Merging and removing the duplicate terms resulted in 303 unique terms closely relevant to the term pedestrian. In the following sections, we refer to these 303 relevant terms as "***augmented terms***."

2.2 Discovering high-level topics: providing a zoom-out view of topics

(Fig. 1-(2)) To better summarize and organize the gathered information and to improve its readability, we intended to extract high-level abstract topics from the "***augmented terms***", acquired in the previous phase. In this step, we experimented with two approaches. While the first approach has not provided us with the expected results, it guided us to a second and improved subsequent approach.

2.2.1 First approach

Preprocessing: To use the root of the "augmented terms", we performed a stemming phase on this set of relevant terms. After stemming the terms back to their basic form, the total number of remaining unique words was 230. Stemming decreased the number of words, as some words ended up having similar root words. For instance, terms such as *cross* and *crossed* have the same root *cross*.

Vectorized Representation: Further, to achieve a set of high-level abstract topics, we aimed to categorize the "augmented terms" into a set of cohesive clusters to place similar terms in the same clusters. To perform clustering, we first transformed each "augmented term" into a vector in the space, using Word2vec models. As explained earlier, the Word2vec algorithm uses simple two-layers neural network models to learn words' associations from a large body of text. Once again, we used the two corpora Google News and Wikipedia to train the neural networks. Once learned, each term was transformed into vectors and located in the space so that terms with higher cosine similarity were placed close to each other.

K-Means Clustering: We then fed these user-defined Word2vec models into a K-Means clustering model [22] to cluster the "augmented terms". K-Means algorithm is an iterative clustering algorithm used to partition the data points into K clusters, whereas each data point is placed in one cluster with the closest mean [32]. Mathematically, the K-Means algorithm is designed to minimize an objective function J. Here we used, the objective function to be the squared error function as below:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^j - \mu_j||^2 \tag{1}$$

where:

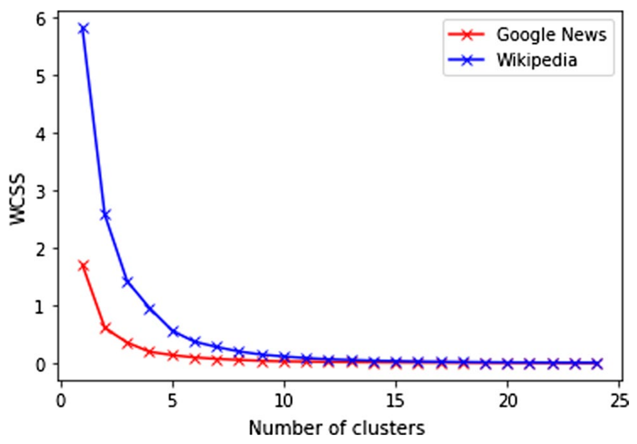

Fig. 4 Elbow method to obtain the optimal number of clusters

J	= objective function
k	= number of clusters
n	= number of cases
x_i^j	= each data point
μ_j	= the centroid for cluster j
$\|\|x_i^j - \mu_j\|\|^2$	= the Euclidean distance

We used the scikit-learn implementation of K-Means clustering, which has slightly different parameters, such as:

- *algorithm:* This parameter allows to select the type of K-Means algorithm to be used. Current research has shown choosing 'elkan' value for this parameter is more efficient on data with well-defined clusters, using the triangle inequality [15]. Thus, we used this value in our experiment.
- *n_clusters:* Representing the number of clusters and the number of centroids to be formed. For this parameter, we selected an optimal number of clusters according to a method explained in the following section.
- *init:* Representing the various methods used for initialization. We used the default method k-means++ [1].
- *n_init:* Indicating the number of times the algorithm should run for different centroids values. We used the default value 10.
- *max_iter:* This parameter indicates the maximum number of iterations for the algorithm. We used the default value of 300.

Optimal Number of Clusters: To determine the optimal number of clusters, we adopted a method called elbow [47]. The elbow method is a technique that plots the relationship between several clusters, used in building the K-Means clustering model, and a chosen metric. Using this method, one can select the elbow of the curve, the point where the curve bends, as the cutoff point, and the optimal number of clusters according to the chosen metric. Choosing the number of clusters exceeding this point results in over-fitting rather than improving the information gained from the clusters. We selected

Within Cluster Sum of Squares (WCSS) as our determining metric. WCSS calculates the sum of squares of each data point's distances in all clusters to their respective centroids. In other words, WCSS measures and represents the variability of the observations within each cluster. In general, a cluster with a smaller sum of squares is more compact than a cluster with a larger sum of squares. As demonstrated in Fig. 4, the optimal number of clusters is five for the Wikipedia Word2vec model and four clusters for the Google News Word2vec model.

Discussion: After selecting the proper number of clusters for the K-Means clustering algorithm, we expected that the group of terms within each cluster to represent a meaningful high-level topic. However, following a manual inspection, all the authors unanimously agreed that the majority of the clustered words were not explicitly relevant to the same high-level topic. This turned the process of labeling clusters with only one meaningful topic, which is relevant to the majority of the terms within the cluster, into an impractical task. For example, one of the clusters contained words, such as *access*, *arcade*, *area*, *bike*, *block*, *bus-only*, *bustling*, *carriageway*, *cobblestone*, *corridor*, *creating*, *east-west*, *elderly*, *front*, *gateless*, *interurban*, *intoxicated*, and *jaywalking*. Since we found the task of obtaining a representative topic for the formed clusters to be impractical, we concluded that this is not the proper approach to form a series of meaningful and representative high-level topics from the "augmented terms". One possible reason for this is that we performed clustering only on individual terms and outside any sentence or paragraph. Clustering words out of their context caused the words to lose their semantics. Thus, we decided to experiment with a slightly different approach, as will be explained next.

2.2.2 Second approach

As an alternative strategy, we formed clusters not merely based on unconnected terms but rather on full sentences, containing additional connecting and supporting terms. Since we are attempting to ultimately specify socially constructed concepts, we used a social media platform, namely *Twitter*, to extract additional relevant text for each "augmented term". We collected *tweets* rather than posts on alternative social media, such as Facebook and Instagram, because Twitter is primarily used for sharing ideas and thoughts, whereas other social platforms are more adopted to connect with friends and family members. To achieve high-level meaningful topics relevant to terms associated with the concept '*pedestrian*', we followed the following steps:

Extracting relevant text (Fig. 2(2)-(a)): To extract tweets relevant to the concept '*pedestrian,*' we used a social media analytics platform called *Crimson Hexagon*[4]. We used Crimson Hexagon to query the Twitter platform for tweets containing terms relevant to the concept '*pedestrian*'. The tool extracts and returns tweets that include the given search query. To extract tweets associated with the term pedestrian, we queried Crimson Hexagon repeatedly, using search queries of a combination of term pedestrian with each of the "augmented terms", such as "*pedestrian and traffic*," "*pedestrian and kill*," "*pedestrian and wheelchair*". Since the number of retrieved tweets for each of these queries varied, the acquired text data were, therefore, imbalanced with respect to queries. The imbalanced dataset could potentially cause a bias in the process of building topics. Therefore, we adopted a sampling approach to select a balanced number of tweets in each group randomly. In case of queries where the returned results were more than 10,000 tweets, we randomly selected 10,000 tweets posted from May 2015 to May 2020. In the case of queries, where the returned result was less than 10,000 tweets, we stretched the start of the time frame to May of 2008. The older tweets, posted before May 2008, were not retrieved for any of the queries to avoid the risk of incorporating outdated data into the semantic web. For the 230 queries, the total number of obtained tweets was 1,326,488. The average number of tweets per query was 5,869, while the minimum number of retrieved tweets was one tweet for the query "*pedestrian and stumbler*". One hundred of the queries collected 10,000 tweets, the maximum number we set. Four of the queries did not return any tweets, including "*pedestrian and gateless*", "*pedestrian and midfoot*," "*pedestrian and stair*," "*pedestrian and stalker*".

Preprocessing the acquired text data (Fig. 2(2)-(b)): After retrieving the raw tweets, we performed preprocessing, by removing hyperlinks, usernames, stop words, punctuation, extra spacing, and numbers from the obtained tweets during this process. Before performing the topic modeling, we performed stemming using the NLTK python library. In stemming, we eliminate the prefixes and suffixes from the inflated terms.

Performing Topic Modeling (Fig. 2(2)-(c)): To perform topic modeling, we adopted the Latent Dirichlet Allocation (LDA) [4]. LDA is a probabilistic approach used to discover latent topics relevant to a set of words in a collection of documents. Each topic yields its own probability distribution over the words, and LDA uses maximum likelihood estimation to learn the best probability distributions for each latent topic. It also explicitly surfaces the probability distribution of words for each learned topic. Because LDA looks to assign topics to cohesive units of text, the latent topics it

[4] https://www.brandwatch.com/.

Table 3 A few of the selected topics for the terms returned by the topic modeling algorithm on the Twitter dataset. Weights are excluded

#	Topic	Top words in the Topic
1	Car killing a person	Hit, car, kill, polic, driver, away, elderli, injur, morn, run.
2	Uber accident	Drive, car, accid, kill, self, uber, fatal, vehicl, driver, hit.
3	Planning and improving safety	Safeti, improv, bicycl, plan, citi, bike, transport, street, project, new.
4	Collapsing bridge	Bridg, one, collaps, sign, way, person, univers, florida, peopl, traffic.
5	Bike & cyclist traffic	Like, good, infrastructur, rule, know, look, traffic, cyclist, lane, bike.
6	Passing the red light	Light, speed, turn, red, cross, limit, left, traffic, driver, car.
7	Traffic signals at junctions	Cross, road, stop, traffic, pleas, use, crosswalk, light, signal, driver.
8	Delay in tunnel causing traffic	Delay, tunnel, expect, bridg, north, traffic, caus, rush, america.

learns often correspond to semantically meaningful topics that can be named.

We used the Scikit-learn library to build topic models from the retrieved tweets. After experimenting with a couple of different values for parameters, the parameters we selected for base topic models were 30 and 25, respectively, for the *number of topics* and *number of words* within each topic. This step's output was 30 clusters, where each cluster was a combination of a different set of words associated with each other representing a topic. Each word is associated with a weight that indicates the importance of that word in a cluster. Table 3 shows a sample output of this step. We excluded the weights associated with each word for representational purposes. From the table, we observe that several words are grouped in each cluster that constitutes a particular topic. For instance, words such as *hit, car, kill, polic, driver, away, elderli, injur, morn, run* were grouped to form a potential topic, such as "*car killing a person*." Similarly, other topics were formed, such as "*planning and improving safety*" and "*collapsing bridge*."

Obtaining topics (Fig. 2(2)-(d), (e)): Through building the LDA model and clustering similar words together, we were able to manually identify a topic for each cluster so that most of the terms in the cluster were relevant to that topic. Table 3 shows some of the topics that we manually derived from the clustered terms. The 30 topics include *"car killing a person"*, *"Uber accident*'', *"planning and improving safety"*, *"collapsing bridge"* , *"bike and cyclist traffic"*, *"passing the red signal"*, *"detecting traffic in Portland"*, *"truck on overpass highway"*, *"space needed on streets of Times Square"*, *"giving access to bridge for traffic"*, *"calling ambulance"*, *"people riding bike"*, *"malls and shops in subway"*, *"pedestrian running at night"*, *"drunk driver kills old man"*, *"police writing ticket to car driver"*, *"jaywalking on flyover"*, *"disabled person needing help to cross"*, *"closed lane of train causing fatal accident"*, *"body in hospital"*, *"tracking sidewalk, walkway and crosswalk"*, *"police arresting driver"*, *"restriction on vehicular movement"*, '*"regularly jog and exercise"*, *"serious injury in vehicle collision"*, *"new bridge construction over river"*, *"parking space on street"*, *"criminal gangs on street"*, *"traffic signals at junctions and intersections"*, *"delay in tunnel causing traffic"*.

2.3 Extending the topics: providing a zoom-in view of topics

(Fig. 1-(3)) In the previous sections, we partially specified the hard-to-specify concept '*pedestrian*' by identifying a set of high-level topics and their associating terms relevant to the concept '*pedestrian*'. Here, we also provide a more detailed or zoom-in view of topics. For this purpose, we need to identify potential relationships among the "augmented terms" and topics retrieved by LDA. For instance, we specified "***wheelchair***" as one of the "augmented terms", closely associated with the concept '*pedestrian*'. We found this term within the LDA clustered terms: {***wheelchair***, user,disabl, need, push, person, barrier, help...}, which we labeled with the topic "***disabled person needing help to cross***". In this step, we intend to specify a more detailed relationship between the "augmented term" "*wheelchair*" and the topic "*disabled person*".

To achieve this, we applied Part-Of-Speech (POS) tagger to the collection of the previously extracted *tweets*. POS tagger reads a sentence and tags each term with their role in the given sentence, such as noun, verb, and adjective. POS taggers generally tag terms in a sentence based on their definition and the context, as corresponding to a particular part of speech.

For the implementation, we used *Stanford CoreNLP*[5] to tag parts of the *tweets* that we previously extracted. We wrote a Python wrapper script that uses the Stanford CoreNLP parser to parse and tag 1,326,488 *tweets*. We labeled the terms in each sentence of the retrieved *tweets* as **triplets** of *subject*, *predicate*, and *object*. Not every sentence necessarily yields a set of triplets. For instance, "*person killed in pedestrian accident near collier exit*" returns "*person, killed, exit*" as the subject, predict, and object, respectively, whereas "*contracting opp wildwood trail pedestrian bridge*

[5] https://stanfordnlp.github.io/CoreNLP/.

project" only returns "*contracting*" as predicate without yielding any subject or object. Similarly, after parsing and probing for triplets in 1,326,488 tweets, we observed that only 1,213,178 tweets yielded triplets.

We further filtered and selected only those triplets in which either the subject or object was among the "augmented terms", resulted in 33,230 triplets. We merged all the predicates, which belonged to the same *subject-object* pair, as a single list. The duplicates having interchangeable *subject-object* pairs were removed. After merging and removing the duplicates, the final result of this process was 333 records. Each record contained a *subject*, an *object*, and a list of all predicates associated with the subject and object. For instance, in one record, we had *pedestrian* as the *subject*, *freeway* as the *object*, and {*killed*, *hit*, *run*, *closed*, *cross*, *identified*, *filmed*, *dies*, *struck*, *call*, *crossing*, *backed*, *injured*, *forcing*} as a set of *predicates*.

In *subject-object* pairs with more than 20 predicates, we filtered the predicates based on similarity scores of the predicate to both the subject and object, using Word2vec models and Wikipedia and Google News corpora as the reference. Therefore, we selected the top 10 predicates with higher similarity scores to the corresponding subject and object.

Finally, to connect the zoom-in and zoom-out views, we selected triplets with subjects or objects matching the "augmented terms". This step's result was 124 triplets to be included in the zoom-in view of the closely related terms to the concept '*pedestrian*'. For instance, in the example at the beginning of this section, the topic "***disabled users***" is a high-level topic associated with the concept '*pedestrian*'. Under this topic, the word "***wheelchair***" is presented as a term (among other sets of words) relevant to the concept and topic. A more detailed zoom-in view represents the Subject-Predicate-Object triplets *{woman parked wheelchair}* and *{bus hit wheelchair}*, where *bus* and *women* are also included within the "augmented terms", closely relevant to the concept '*pedestrian*'. In the following section, we discuss how we represent these terms and relationships in the form of a semantic web. Further, we explain how we use the constructed web for the hard-to-specify concept '*pedestrian*' to improve a state-of-the-art object detector's accuracy.

2.4 Representation

(Fig. 1-(4)) The goal of a semantic web is to make internet data machine-readable [13]. *Controlled Natural Languages (CNLs)*, such as Web Ontology Language (OWL), are used to express knowledge about resources in a semantic web and specify relations in a human-readable way [6, 48]. The benefit of CNLs is that while they are understandable by humans (i.e., declarative language), they can be directly translated to a formal language (i.e., have formal syntax).

We exploit the terms and relations that we extracted in the previous steps to build a semantic web for the hard-to-specify domain concept '*pedestrian*'. In the semantic web, an *entity* was formed for each distinct term; and an edge was drawn between two nodes for each specified relation. This process resulted in 78 entities for unique terms of subjects and objects, as well as 339 unique edges representing predicates terms in the semantic graph. Our semantic web benchmarks a common understanding of the concept and facilitates better communication over domain knowledge between humans and MLCs. This benchmark represents the shared and consensual knowledge of domain specifications, which is generally accepted by the public.

To visualize the selected domain concept, we used a tool called WebVOWL[6] (Web-based Visualization of Ontologies). WebVOWL uses a Resource Description Framework (RDF) file as the input to visually display the terms and their relations. We wrote a script to automatically transform the subject, object, and predicate triplets into an acceptable format for the tool. Due to the large size of the generated semantic web, we only provided a snapshot of it, as shown in Fig. 5. The complete semantic web can be found on our GitHub repository[7]. In the following sections, we discuss how we exploited the established benchmark as a reference for augmenting the concept '*pedestrian*' in a collected dataset to improve the accuracy of an object detector (MLC).

3 Augmenting the dataset

This section discusses our evaluation of the proposed approach. Assessing *correctness* of the constructed semantic web through conducting a qualitative study with human participants is highly subjective to the participants' conditions, experiences, and perspectives and, therefore, can be biased. In addition, evaluating *completeness* of the built semantic web in specifying *all* variances of the concept, pedestrian, is inconclusive. In view of the raised issues, to instead evaluate the *usefulness* of the proposed approach we (i) first, evaluated the approach ***effectiveness*** in improving the accuracy of MLCs. For this purpose, we identified the existing gap in a state-of-the-art pedestrian dataset and augmented it for one missing entity. We then compared the accuracy of a state-of-the-art object detector, trained on this dataset, in detecting various instances of '*pedestrian*' before and after the augmentation. (ii) second, we evaluated the ***coverage*** of the generated entities and relations of the semantic web on multiple commonly used datasets, which in turn, the gaps in these datasets.

[6] http://vowl.visualdataweb.org/webvowl.html.

[7] https://github.com/REJournal2021.

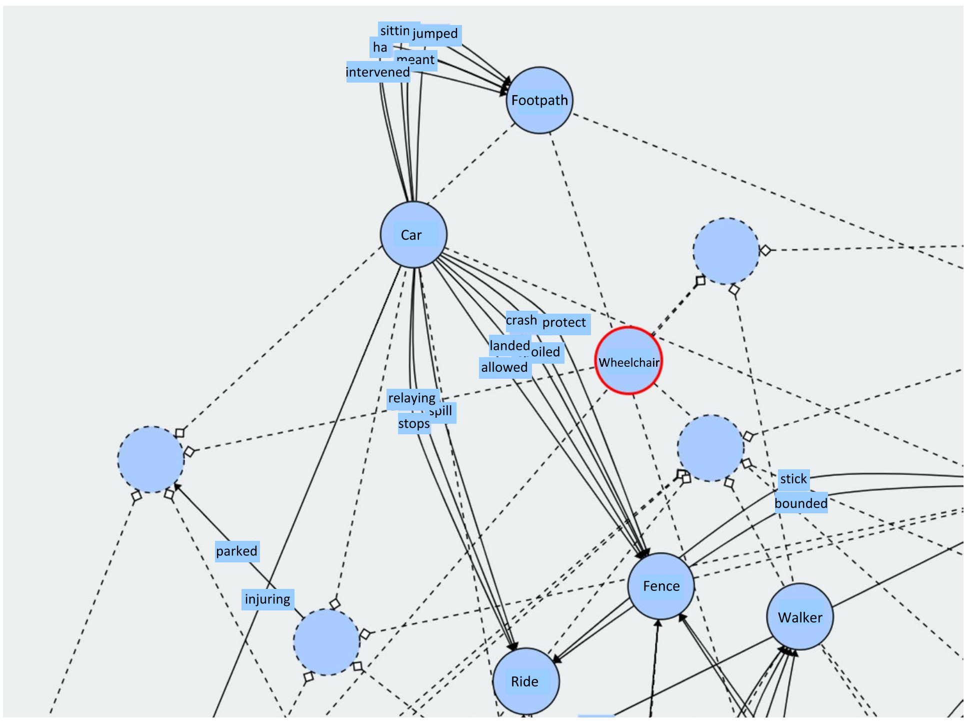

Fig. 5 A snapshot of the constructed semantic web for the hard-to-specify concept '*pedestrian*'. Circles represent the entities, and the labeled edges show the relations between the pairs of entities

3.1 Evaluating effectiveness

To evaluate the effectiveness of the established multi-level semantic web, we initially selected a benchmark dataset. We identified one common and semantically important feature, *wheelchair*, that was often associated with potential pedestrians with respect to the constructed web, (Fig. 1-(5)). In our semantic web wheelchair showed relatively high similarity to the domain concept, and frequent co-appearance with the term pedestrian in corpora, yet it was missing from the images of the primary dataset. Although we selected wheelchair as a missing feature, this decision was not only due to its high semantic similarity to pedestrian, but also our investigations of five state-of-the-art pedestrian and person datasets, which we will further introduce in Sect. 3.2.1, revealed the term's absence from 80% of the datasets. Therefore, we manually collected a balanced number (w.r.t. the size of other classes in the dataset) of images containing the instances of the concept (various-looking pedestrians) interacting (sitting in) with the missing feature (wheelchair), (Fig. 1-(6)). We augmented the training dataset and re-trained the ML model on the augmented dataset for a fair comparison of two models' accuracy in classifying pedestrians (Fig. 1-(7)).

Please note that we only augmented the dataset for one partial specification (one triplet in the semantic web) as the image collection, localization, and annotation for the missing feature, in this case, was manually performed and required a significant amount of labor. However, this manual task needs to only occur once because when "enough" number of images—relative to the size of other classes—are collected for an underrepresented feature, then a ML model can be re-trained to classify the missing feature in images. Thus, this new model can be applied to automatically identify a set of images that contain the missing feature among a large collection of images. In addition, we further discuss an alternative method to automate this process in Sects. 3.2.2 and 3.2.3 .

Primary Dataset: We used images from the dataset Microsoft Common Objects in Context (COCO) [31]. This dataset is a publicly available large-scale collection of common objects' images in their natural context. Each image is annotated with all objects recognizable in the image through crowd workers' manual collaborative work. Additionally, the objects' boundaries are identified and segmented in each image. Image annotations and segmentations are further analyzed and evaluated by a group of expert workers. COCO consists of 2.5 million labeled instances in 328k images, labeled with 91 different object categories [31]. We selected this dataset because more than 35% of annotated objects are 'person' with various poses, characteristics, and within different contexts. Moreover, the COCO dataset contains more categories and instances per image on average than other publicly available datasets, such as PASCAL and ImageNet. Additionally, YOLO, a state-of-the-art real-time object detector, is pre-trained and holds weights learned according to the COCO dataset.

We decided to conduct a series of experiments to evaluate the application of our semantic web. Due to the large size of images in COCO, using the entire dataset resulted in

Table 4 Statistics of the selected dataset from COCO using stratified random sampling

	Object-no	Image-no	Image-percent	Object-percent	Obj/image
Person	27518.0	5139.0	72.62	26.15	5.35
Chair	7138.0	2243.0	31.69	6.78	3.18
Car	6604.0	1710.0	24.16	6.28	3.86
Bottle	4695.0	1693.0	23.92	4.46	2.77
Book	4225.0	916.0	12.94	4.01	4.61
Cup	4017.0	1646.0	23.26	3.82	2.44
Bowl	2440.0	1165.0	16.46	2.32	2.09
Handbag	2417.0	1329.0	18.78	2.30	1.82
Dining table	2149.0	1489.0	21.04	2.04	1.44
Backpack	1788.0	1133.0	16.01	1.70	1.58
Bench	1585.0	852.0	12.04	1.51	1.86
Knife	1572.0	674.0	9.52	1.49	2.33
Truck	1532.0	921.0	13.01	1.46	1.66
Traffic light	1481.0	484.0	6.84	1.41	3.06
Potted plant	1461.0	784.0	11.08	1.39	1.86
Umbrella	1166.0	444.0	6.27	1.11	2.63
Spoon	1152.0	613.0	8.66	1.09	1.88
Bicycle	1116.0	545.0	7.70	1.06	2.05
Wine glass	1106.0	363.0	5.13	1.05	3.05
Sink	1049.0	902.0	12.75	1.00	1.16
Tv	933.0	716.0	10.12	0.89	1.30
Fork	893.0	498.0	7.04	0.85	1.79
Suitcase	803.0	389.0	5.50	0.76	2.06
Vase	767.0	465.0	6.57	0.73	1.65
Laptop	760.0	564.0	7.97	0.72	1.35
Couch	738.0	581.0	8.21	0.70	1.27
Cell phone	702.0	573.0	8.10	0.67	1.23
Sports ball	677.0	462.0	6.53	0.64	1.47
Apple	614.0	220.0	3.11	0.58	2.79
Carrot	596.0	179.0	2.53	0.57	3.33
Oven	580.0	502.0	7.09	0.55	1.16
Orange	550.0	209.0	2.95	0.52	2.63
Boat	544.0	182.0	2.57	0.52	2.99
Motorcycle	552.0	230.0	3.25	0.52	2.40
Remote	535.0	322.0	4.55	0.51	1.66
Bus	536.0	390.0	5.51	0.51	1.37
Keyboard	527.0	374.0	5.28	0.50	1.41
Bird	521.0	166.0	2.35	0.50	3.14
Dog	487.0	371.0	5.24	0.46	1.31
Mouse	482.0	397.0	5.61	0.46	1.21
Kite	471.0	102.0	1.44	0.45	4.62
Cake	470.0	208.0	2.94	0.45	2.26
Banana	461.0	227.0	3.21	0.44	2.03
Refrigerator	456.0	434.0	6.13	0.43	1.05
Clock	453.0	402.0	5.68	0.43	1.13
Microwave	449.0	428.0	6.05	0.43	1.05
Toothbrush	443.0	238.0	3.36	0.42	1.86
Baseball bat	435.0	303.0	4.28	0.41	1.44
Horse	418.0	216.0	3.05	0.40	1.94
Donut	420.0	98.0	1.38	0.40	4.29
Cow	418.0	159.0	2.25	0.40	2.63

Table 4 (continued)

	Object-no	Image-no	Image-percent	Object-percent	Obj/image
Sandwich	416.0	209.0	2.95	0.40	1.99
Skis	417.0	160.0	2.26	0.40	2.61
Elephant	406.0	187.0	2.64	0.39	2.17
Sheep	410.0	77.0	1.09	0.39	5.32
Pizza	411.0	178.0	2.52	0.39	2.31
Parking meter	409.0	230.0	3.25	0.39	1.78
Teddy bear	410.0	241.0	3.41	0.39	1.70
Skateboard	408.0	233.0	3.29	0.39	1.75
Scissors	400.0	301.0	4.25	0.38	1.33
Toilet	400.0	348.0	4.92	0.38	1.15
Broccoli	403.0	166.0	2.35	0.38	2.43
Airplane	400.0	234.0	3.31	0.38	1.71
Bed	402.0	359.0	5.07	0.38	1.12
Bear	400.0	291.0	4.11	0.38	1.37
Zebra	402.0	158.0	2.23	0.38	2.54
Giraffe	400.0	225.0	3.18	0.38	1.78
Tie	404.0	238.0	3.36	0.38	1.70
Frisbee	400.0	282.0	3.98	0.38	1.42
Snowboard	403.0	206.0	2.91	0.38	1.96
Baseball glove	403.0	282.0	3.98	0.38	1.43
Surfboard	400.0	172.0	2.43	0.38	2.33
Tennis racket	400.0	254.0	3.59	0.38	1.57
Cat	400.0	348.0	4.92	0.38	1.15
Stop sign	402.0	370.0	5.23	0.38	1.09
Fire hydrant	403.0	372.0	5.26	0.38	1.08
Hot dog	402.0	147.0	2.08	0.38	2.73
Train	403.0	337.0	4.76	0.38	1.20
Toaster	225.0	217.0	3.07	0.21	1.04
Hair drier	198.0	189.0	2.67	0.19	1.05

excessively expensive experiments and therefore limited the number of experiments we were willing to perform. For this reason, we instead systematically selected a subset of COCO to perform a variety of experiments. To select our samples, we used a *stratified random sampling* technique rather than simple random sampling to represent the population of images in each category. In stratified sampling, data points are randomly selected from each sub-population (category) in proportion to its original size. Samples in each category (i.e., strata) are expected to share the same characteristics. In our case, the samples in each category share the same class label. As a result, more samples were selected from categories with a larger number of images, and similarly, a smaller number of images were collected from the smaller groups. Table 4 represents our dataset's statistics, including 7077 total images and total objects labels of 105,239 selected from the COCO dataset. For ease of reference, we refer to this selected subset of COCO as COCO in the rest of this article and Table 5.

Auxiliary Datasets: Moreover, we used two additional datasets, namely Open Images and Mobility Aids datasets, to retrieve images of entities that were missing from the primary dataset according to our semantic web. Open images are a dataset that has approximately 9 million varied images with rich annotations[8]. The images are very diverse and often contain complex scenes with several objects (8.4 per image on average). It contains 15,851,536 boxes on 600 categories, image-level labels annotations, object bounding boxes, object segmentation, visual relationships, and localized narratives.

Object Detector: In our experiments, the YOLOv5 algorithm was used for object detection, specifically to detect pedestrians (people in the context of streets) in the image dataset. Since the COCO dataset contains people's images in the streets, we used the existing YOLO's detectable label '*person*' to be interchangeable with '*pedestrian*'. The

[8] https://storage.googleapis.com/openimages/web/index.html.

process of object detection in YOLO happens through two levels, involving image classification and object localization. Image classification assigns an image to a different set of existing classes or categories, such as a person, car, and plane, whereas object localization positions the objects in the image. Through this process, the object detector algorithm identifies objects and their positions in an image by labeling the objects and creating a *bounding box* around them. The object detector also generates a *confidence score*, representing the probability that an anchor box contains an object. The bounding boxes are later compared to the ground-truth according to an evaluation metric called Intersection over Union (IoU). IoU score identifies the area of overlap between the predicted bounding box and the ground-truth bounding box. According to the IoU value of interest, the classification accuracy is computed according to standard metrics such as Recall, Precision, and Mean Average Percentage (mAP) score (with different IoU values). The algorithm considers a detected label to be correct if the detected class matches the label in the ground-truth, the confidence score is greater than the selected threshold, and the detected bounding box for the object overlaps with X% of the bounding box in the ground-truth, where X is the selected value for IoU. In our experiments, we selected X to be 50%, and in another case, range from 50% to 95% (Table 7).

Experiments: Here, we explain the series of experiments we conducted to evaluate the application of our previously constructed semantic web. Table 5 represents a summary of all our experiments, while Table 7 reports the object detector's performance in the conducted experiments for detecting the label '*Person*'. To better evaluate our proposed approach for building the semantic web, we divided the experiments into two main categories (1) *Before Dataset Augmentation* and (2) *After Dataset Augmentation*.

In the experiments within the *Before Dataset Augmentation* category, we recorded YOLO's performance on our selected dataset, **before** referring to the constructed semantic web for dataset augmentation. These sets of experiments served as base cases for assessing YOLO's improvements after augmenting the dataset according to the semantic web.

After Dataset Augmentation category consists of experiments for evaluating YOLO's performance **after** we augment the dataset with reference to our semantic web. For this purpose, we carefully went through the primary dataset as well as the set of subject-predicate-object triplets, obtained in section 2.3, representing entities with close relations to the pedestrian. We identified a set of subjects and objects that were not initially present in the collection of pedestrian images within the primary dataset. To augment the dataset, we chose one missing object or subject at a time and added a balanced number of images of the subject or object to the original dataset. These additional images were selected from our auxiliary datasets, Open Images, and Mobility Aids.

Augmented Training Datasets: In all of our experiments in this article, within *After Dataset Augmentation* category, we considered *wheelchair* as a missing object from the primary dataset. We chose the wheelchair because it appeared in our semantic web subject-predicate-object triplets, showing a strong association with the term '*pedestrian*' and was not initially present in the primary dataset. We selected 608 images of *pedestrians on wheelchairs* from the Open Image and Mobility Aids datasets for training purposes. For the missing term wheelchair, we augmented two versions of training datasets with additional images:

1. **Merged-Labels Dataset:** In this version, as shown in Fig. 6b, we augmented the dataset with the images of *pedestrians on wheelchairs*, where the bounding box is drawn around the entire boundary of the person and wheelchair together. We selected the wheelchair and the person in the wheelchair as one object, and labeled them as '*Person*', and created a bounding box around them. We augmented the COCO dataset with this set of labeled images, which resulted in 117,266 and 4952 images, respectively, for training and validation.
2. **Split-Labels Dataset:** In the second version of the augmented dataset, as illustrated in Fig. 6c, d, using the same set of 608 images of pedestrians on wheelchairs, we created two different bounding boxes: one around the person and one around the wheelchair. We labeled the two objects with two separate labels as '*Person*' and '*Wheelchair*'.

The primary reason for conducting experiments with two training datasets is to study the impact of augmenting the dataset with both a new label, '*Person-in-wheelchair*', and an already existing label, '*Person*'.

Testing Datasets: We evaluated the performance of YOLO on two testing datasets.

1. **Basic Dataset:** In the first version, we randomly selected 106 images containing a pedestrian in a wheelchair from Open Images and Mobility Aids datasets. We ensured that these 106 images were not included in our training datasets and were previously unseen by the object detector. This dataset is to evaluate YOLO's performance on unseen data after the dataset augmentation.
2. **Integrated Dataset:** In the second version, we integrated the same 106 images of the *Basic Dataset* into our primary dataset to evaluate the overall performance of YOLO on previously seen images in addition to the unseen ones. In three experiments, we only used the COCO dataset as our testing set to evaluate YOLO only on previously seen data.

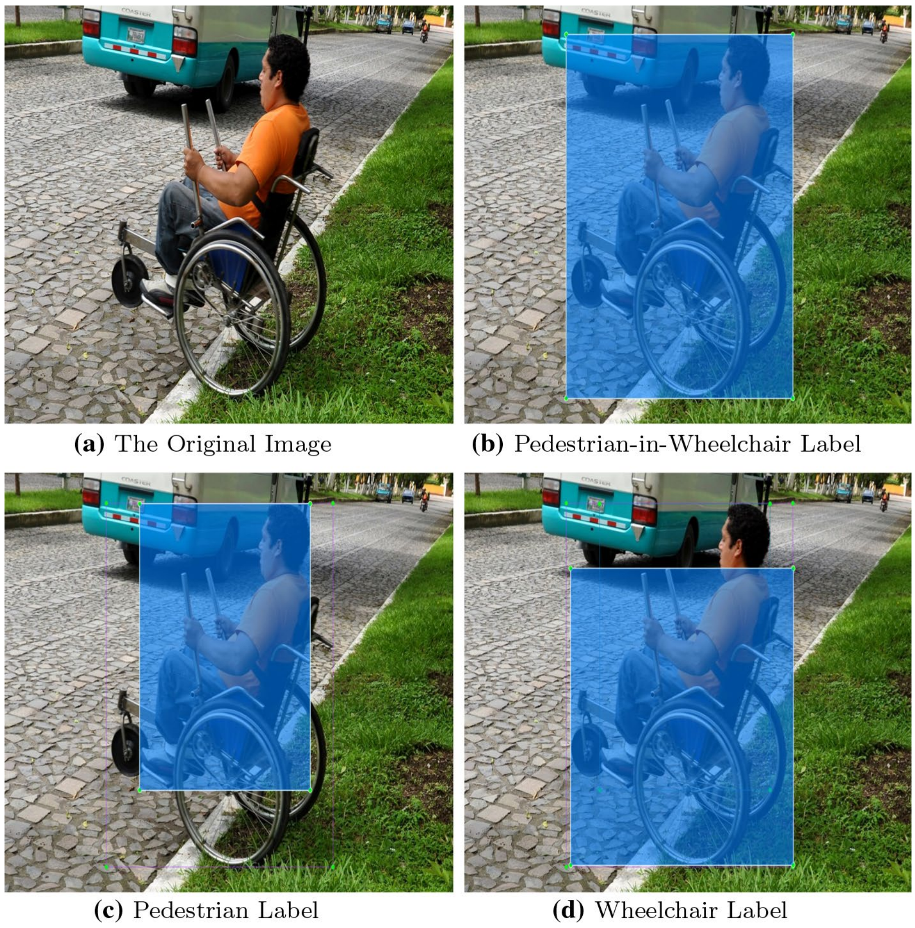

Fig. 6 Different labels for a sample image

Table 5 A summary of the conducted experiments

	Exp.	Training set	Testing set	Ground truth/added label
Before Dataset Augmentation (Base Case)	1a	-	Integrated	Merged-Labels
	2a	-	Integrated	Split-Labels
	3a	-	Basic	Merged-Labels
	4a	-	Basic	Split-Labels
	5a	-	COCO	Default
After Dataset Augmentation	1b	Merged-Labels	Integrated	Person-in-wheelchair
	2b	Split-Labels	Integrated	Person + Wheelchair
	3b	Merged-Labels	Basic	Person-in-wheelchair
	4b	Split-Labels	Basic	Person + Wheelchair
	5b	Merged-Labels	COCO	Person
	5c	Split-Labels	COCO	Person + Wheelchair

In all the experiments, we only report YOLO's performance on detecting the label '*Person*' as our dataset augmentation process is only concerned with this label and does not interfere with other classes' trained weights.

3.1.1 Before dataset augmentation (base cases)

– **Experiment 1a:** In this experiment, we chose the Merged-Labels dataset. We assessed YOLO's overall performance in detecting the label '*Person*' only for the 106 unseen images of pedestrians in a wheelchair in addition to the seen images of COCO (Integrated dataset). Note that the ground-truth for pedestrians' images

Table 6 YOLO performance report for detecting label *Person* before and after the augmentation

	Exp.	Recall	Precision	mAP%50	mAP%50-to-%95
Before (Base Cases) Dataset Augmentation	1a	0.749	0.311	0.585	0.327
	2a	0.935	0.162	0.300	0.163
	3a	0.872	0.153	0.249	0.116
	4a	0.751	0.311	0.591	0.331
	5a	0.745	0.324	0.630	0.354
After Dataset Augmentation	1b	0.747	0.327	0.627	0.351
	2b	0.957	0.485	0.898	0.626
	3b	0.943	0.764	0.934	0.707
	4b	0.738	0.349	0.635	0.366
	5b	0.747	0.329	0.630	0.352
	5c	0.732	0.344	0.629	0.357

Experiments match their baseline with the same color

Table 7 Dataset statistics: object and relations are detected by USGG

Dataset	#Image	#Object	#Distinct object	#Relation	#Distinct relation
COCO	7077	212,310	149	141,540	27
CrowdH.	15,000	450,000	148	300,000	26
WiderP.	8000	240,000	147	160,000	19
CityP.	2975	89,250	102	59,500	17
Caltech	4285	128,550	111	85,700	22
ECP	23,892	716,790	134	477,860	22

in a wheelchair in this experiment was the Merged-Labels dataset and, therefore, had the bounding box, as shown in Fig. 6b.

- **Experiment 2a:** In this experiment, similar to the previous experiment, we assessed YOLO's overall performance in detecting the label '*Person*' for images in the Integrated dataset. The only difference is that in this experiment, the ground-truth was the Split-Labels dataset. Therefore, the images of pedestrians in a wheelchair had the bounding box as illustrated in Fig. 6c, d.
- **Experiment 3a:** In experiment 3a, we repeated experiment 1a; however, we evaluated YOLO's performance only on the 106 images of pedestrians in a wheelchair (Basic dataset). The ground-truth in this experiment similarly was the Merged-Labels dataset.
- **Experiment 4a:** In experiment 4a, similar to experiment 3a, we assessed YOLO's performance in detecting the label '*Person*' on the Basic dataset. The only difference is that in this experiment, the ground-truth was the Split-Labels dataset.
- **Experiment 5a:** In this experiment, we ran the original pre-trained YOLO on our selected subset of COCO.

3.1.2 After dataset augmentation

- **Experiment 1b:** In this experiment, using the Merged-Labels dataset contained pedestrians in wheelchair images, we retrained YOLO with a new label '*Person-in-wheelchair*' for 20 epochs with pre-trained checkpoints as the initial model weights. We evaluated YOLO's performance on the Integrated dataset.
- **Experiment 2b:** Here, we repeated the previous experiment using the Split-Labels dataset instead of retraining YOLO with the new label '*Person-in-wheelchair*', we retrained it with the existing label '*Person*' and a new label '*Wheelchair*'. In this experiment, we tested YOLO's overall performance on the Integrated dataset.
- **Experiment 3b:** In this experiment, similar to experiment 1b, we once again used the Merged-Labels dataset and re-trained YOLO with the new label '*Person-in-wheelchair*'. We evaluated YOLO's performance on the Basic dataset of pedestrians in a wheelchair.
- **Experiment 4b:** This experiment is the repetition of the previous experiment, except that in this experiment, we re-trained the model with the Split-Labels dataset.
- **Experiment 5b:** In this experiment, we applied the same re-trained model in experiment 3b, but we tested the model only on the COCO dataset.
- **Experiment 5c:** In our last experiment, we applied the same re-trained model in experiment 4b, but we tested the model only on the COCO dataset (Table 6).

3.1.3 Comparing results and discussion

To compare results and conclusions more expressive, we discuss each experiment's results by comparing it with its associated baseline. The association between the experiments and their relevant base cases are also color-coded in Table 7. For instance, experiment 2a serves as the baseline for experiment 2b, and both demonstrated in blue. Similarly, Experiment 3a serves as the baseline for experiment 3b (in green), while experiment 4a is the base case for experiment 4b (in red).

Overall Performance:

- **After Adding '*Person-in-wheelchair*' Label:** To assess the change in YOLO's overall performance on seen and unseen images after re-training the model with the Merged-Labels dataset, we compare the accuracy metrics of experiments 1a and 1b. As shown in table 7, we observe a slight improvement in precision and mAP values. The mAP %50 is increased from 0.585 to 0.627 and mAP %50-to-%95 from 0.327 to 0.351.
- **After Adding '*Person*' and '*Wheelchair*' Labels:** We observed an increase of mAP %50 value from 0.300 in experiment 2a to 0.898 in experiment 2b, where we re-trained YOLO using two separate labels of '*Person*' and '*Wheelchair*'. Additionally, recall is increased from 0.935 to 0.957.

Performance on Unseen Data:

- **After Adding '*Person-in-wheelchair*' Label:** To evaluate YOLO's performance before and after being retrained with the new label and over a set of images that the model has not seen before, we compare the results of experiments 3a and 3b. As reflected in Table 7, the recall and precision are improved from 0.872 and 0.153 to 0.943 and 0.764, respectively. Similarly, mAP values for 50% and 50%-to-95% IoU are increased from 0.249 and 0.116 to 0.934 and 0.707.
- **After Adding '*Person*' and '*Wheelchair*' Labels:** The comparison of experiments 4a and 4b represent the change in YOLO performance in detecting pedestrians before and after the model was retrained with Split-Labels dataset. We observe an improvement in precision from 0.311 to 0.349. The mAP %50 and mAp %50-to-%95 values are increased from 0.591 and 0.331 to 0.635 and 0.366, respectively.

Performance on Previously Seen Data:

- **After Adding '*Person*' Label:** For the evaluation of YOLO only on the COCO dataset after retraining the model with the Merged-Labels dataset, we compare experiments 5a and 5b. As illustrated in Table 7, recall and precision are slightly improved from 0.745 and 0.324 to 0.747 and 0.329.
- **After Adding '*Person*' and '*Wheelchair*' Labels:** We observed a slight decrease in recall value from 0.745 in experiment 5a to 0.732 in experiments 5c and a slight increase of precision value from 0.324 to 0.344.

To summarize, augmenting the dataset according to our semantic web (for the missing feature '*wheelchair*') improved the precision and mAP %50 values in all the conducted experiments. In a few of the experiments, this improvement appeared to be minor. However, considering the safety context of the problem at hand, even slight performance improvements for pedestrian detection can save lives. Additionally, to the best of our knowledge, our proposed approach has the novelty to provide the foundation of a pragmatic approach for addressing the inherent ambiguity of domain concepts for MLCs.

3.2 Evaluating coverage

Since evaluating the completeness of the established terms and relationships—within the semantic web—in specifying every variance of potential pedestrians is somewhat inconclusive, we instead assessed the semantic web's relative completeness with respect to several state-of-the-art pedestrian, human, and generic datasets. For each dataset, we first translated the contained images into natural language; then we found the intersection of image descriptions with entities and relations established within our semantic graph. Evaluating the coverage of "important" features of a pedestrian and their relations—that the proposed approach identified—in commonly-used datasets, provides an empirical estimation of whether the approach returns generally useful specifications for the domain concept pedestrian.

3.2.1 Datasets

For this purpose, in addition to COCO dataset [31], previously introduced, we referred to three of the most commonly-used datasets of pedestrian images and video frames, namely Caltech [14], CityPersons [53], and EuroCity Persons (ECP) [8]. The pedestrian dataset benchmarks have been proposed from the context of autonomous driving. However, these datasets are monotonous, such that they lack diverse scenarios. Hence, we selected two additional large-scale human datasets, CrowdHuman [40], and WiderPerson [54], which unlike pedestrian datasets, are not limited to traffic scenarios, and include images of people in more generic contexts, such as people in parks, restaurants, and selfies.

3.2.2 Converting image datasets to natural language

Since the entities and relations in the generated semantic web are in natural language, to evaluate the graph coverage, we initially translated the image datasets to natural language as well. As such, we were able to determine whether the semantic web's partial specifications appeared in the image descriptions, and to identify the specifications that are underrepresented in benchmark datasets.

For this purpose, we carefully reviewed and experimented with the existing approaches in Computer Vision (CV) domain. As a result, we selected the state-of-the-art

Unbiased Scene Graph Generation (USGG) framework which generates a scene graph from an image so that to describe the scenes in the image [44]. The model first, detects the objects present in an image, and second, extracts the existing relationships between the detected objects in the form of $Object_1$-Relation-$Object_2$ triplets. The model is previously trained on Genome Dataset across 75,000 object categories and 37,000 relations categories [29].

To the best of our knowledge USGG framework generated less biased and more accurate predictions over other existing approaches that we investigated, such as Iterative Message Passing [51], MOTIFs [43, 52], and VCTree [29]. USGG also provided more fine-grained relationships in comparison to alternative models, such as replacing *near* with *behind/in front of*, and *on* with *standing on/walking on/parking on/driving on*.

Applying USGG to each of the six datasets, we extracted, in total, about 2 and 1.5 millions objects and relations, respectively. Table 7 represents these numbers in more detail for each dataset. As shown, the largest number of objects and relations belong to ECP dataset, while COCO contains the largest number of distinct objects and relations.

3.2.3 Mapping the semantic web to image descriptions

To estimate the coverage of the established semantic web with respect to each dataset, we initially searched for the exact word-to-word matches between the object terms, extracted by USGG, and entities in the constructed semantic web. Similarly, we looked for the exact matches between the established relations for each pair of entities in the web and USGG-extracted relation terms. Word-to-word mapping resulted in an average of about 10 out of 78 and 3 out of 339 matches for entities and relations, respectively.

Further, to additionally identify the terms that were not necessarily word-to-word match, yet referred to the same feature, we also considered semantic similarity between the terms. For this purpose for each dataset, the cosine similarity score, according to Google News Word2vec model, was computed for each entity in our graph with each object that USGG detected in the dataset; similarly between each graph relation and each USGG-detected relation. Later, for each entity and each relation in the web only the top five USGG-detected terms with the highest similarity scores were selected. Among the remaining terms, there were still terms with relatively low similarity scores. Such dissimilar terms with similarity score less than $\mu - \sigma$ were further removed, where μ is the mean of all similarity scores in the dataset; and σ is the standard deviation of the distribution. As such, the remaining term(s)—the number of the terms is between 0 and 5—, represent the most probable matches among the USGG-detected objects for each entity that the proposed approach established in the semantic web. We repeated the same process to semantically map the relations, we inferred and represented on the edges of the semantic web, to relations that USGG detected between the objects in the images.

Table 8 Coverage of the semantic web entities w.r.t. each dataset and USGG

Dataset	Precision	Recall	F1
COCO	0.697	0.524	0.598
CrowdHuman	0.697	0.530	0.602
WiderPerson	**0.710**	0.540	0.613
CityPerson	0.697	**0.630**	**0.662**
Caltech	0.693	0.565	0.622
EuroCity Person	0.684	0.553	0.611

Finally to evaluate the coverage of the established web in each dataset, we asked an independent PhD researcher—with no information about or connection to the project—to label the remaining matched pairs as “meaningful”, “non-meaningful”, or “I don’t know”. We asked the researcher to select “meaningful” if *only according to her intuition* both terms represent the same feature, “non-meaningful” if they do not, and “I don’t know” if she is not certain about their relevance. We did not limit the time of the study and allowed the researcher to complete the task using as much time as necessary. Finally, we computed precision, recall, and F1 measure according to the final labels, considering *(entity, object)* pairs labeled “meaningful” as true positives; *(entity, object)* pairs labeled “non-meaningful” as false positives; and *(object, entity)* pairs labeled “non-meaningful” as false negatives, which were “important” pedestrian features erroneously missing from the web. There was no pair labeled as “I don’t know” in the study showing that the evaluator was certain about the labels she selected. Table 8 represents the results of the evaluation of the semantic web entities with respect to USGG-detected objects in each dataset. As shown, the semantic web has the highest precision with reference to the WiderPerson dataset, and the highest recall and F1 scores in comparison to CityPerson dataset. We discuss the results more in detail in the following subsection.

We only evaluated the coverage of the semantic graph’s entities, representing the “important” features of the concept pedestrian, and not the relations. The reason for excluding the evaluation of relations is that considering their large number in the semantic web—there are multiple inferred and established relations between each pair of entities—, the cross product between the graph relations and the USGG relations was too large to be manually evaluated in a reasonable time. For instance, to evaluate the coverage of the graph entities, a total of 78 unique terms of subject and object are contained in the semantic web. USGG detected 149 unique terms of $Object_1$ and $Object_2$ in the images of COCO dataset. The cross product between them results in 11,622 pairs, which after the removal of less confident pairs this number is reduced to 390 to be manually evaluated. However, this number was 1,580 for evaluating the coverage of the graph relations in the same dataset.

3.2.4 Discussion

According to the evaluation, an entity, for which none of the—most semantically similar—USGG-detected and matched terms is evaluated as "meaningfull" by our human evaluator, potentially represents a non-relevant feature to the concept pedestrian i.e., false positives of the approach. However, such non-matched entities do not necessarily represent a meaningless or an impractical feature, specified by our approach, as there are several other factors involved in the scenario: (1) The entity, in the best case scenario, may propose an underrepresented feature in the dataset. (2) There is also a chance that the USGG detector was not trained, and therefore, was unable to detect the particular object in the images. (3) The semantic similarity algorithm failed to create a semantic mapping between the terms. (4) The corpora which was referred to for establishing the semantic mapping may not contain the appropriate similar terms within its context. (5) The relevant terms that could establish a meaningful mapping were mistakenly eliminated during the process of filtering the less confident terms. While it is difficult to draw a concrete conclusion, the results in table 8 provide an estimation of the semantic web's relative completeness with respect to USGG and the selected datasets.

Further a review of the evaluated mapped pairs, flagged with both "meaningful" and "non-meaningful", we noticed several interesting patterns. For instance, among those labeled as "meaningful" there were several pairs with similar terms referring to the same feature, such as the entities *sidewalk*, *walkway*, and *footpath* in the semantic web that were associated with the object *sidewalk* detected by USGG. In the same order were the entities *gate* and *entrance* with the detected object *door*; *subway* with *train*; *light* with *lamp*; *mall* with *building*, *path* with *track*; *stop sign* with *sign*; and *road* with *street*. However, we also noticed several pairs with similar terms that were not precisely referring to the same feature, yet were relevant to the same subject, such as the entity *tunnel* and *rail* in our web with the object *train* detected by USGG. In the same order *ride* and *horse* with *bike*; and *cyclist* with *bike*.

Another interesting pattern we identified among "non-meaningful" pairs that potentially revealed the possible misclassification candidates. For instance, the entities *walker* and *wheelchair* in the web were semantically mapped with the object *bike* in datasets. While both pairs were evaluated as "non-meaningful" by the researcher, aside from possibly proposing two underrepresented features (wheelchair and walker in the dataset), they could also suggest that the classifier, if not re-trained, may potentially detect wheelchair and walker as bike mistakenly.

Several pairs proposed a corresponding detectable object with perceptual concepts in the semantic web, assisting to explain, add, and precept a concept in an image through the associated object which is detectable by the classifier. For instance, the entities *safety* and *injuries* were mapped to detectable object *helmet*; the entity *drunk* was matched with the detected object *bottle*; and the entities *accident* and *fatality* in the semantic web were mapped to the detected object *motorcycle*.

4 Threats to validity

Our study has several potential threats. Concerning generalizability, we experimented with only one domain, the automotive domain, and one hard-to-specify domain concept, '*pedestrian*'. We selected this domain because of the increasing application of ML in software systems being used in this domain. Additionally, recent extensive testing of Autonomous Vehicle (AV) on public roads has raised serious concerns [9, 10, 26–28], especially with the recent reported failures of autonomous driving systems [34, 46]. Moreover, in the automotive domain, there are significant safety risks that are introduced due to the lack of an explicit set of requirements specifications and the ambiguity of potential specifications. We selected this concept because the term pedestrian is present in many potential requirements specifications relevant to this domain, thus is the root of ambiguity in specifications. To the best of our knowledge, there is no other systematic method to ground hard-to-specify socially-constructed domain specifications for MLCs. Thus, our initiative work and our future extensions can be the start of a significant contribution to this problem in the domain. Additionally, the process we defined is general and fully automatable (with the exception of topic selection) and can be applied to specify other domain-specific concepts in other domains.

The creation of the semantic web is based on the information that we extracted from the web and social media, namely Twitter, Wikipedia, and Google News. These platforms contain accurate, true, and relevant but also inaccurate, false, and out-dated information. Considering the socially-constructed nature of our concepts and the lack of documentation for these concepts, we found social data to be helpful for specifying socially-constructed concepts. To mitigate bias, we only extracted up-to-date information, and we used more real-time idea-sharing platforms. However, further work is needed to evaluate the validity of the extracted information and to claim absolute *completeness* of the constructed semantic web. Further, using Natural Language Processing (NLP) techniques introduces linguistic limitations, such as polysemy, to the approach that is required to be addressed in future work.

Finally, in terms of evaluation, we have only used one dataset and object detector to illustrate the application of our approach. Although the selected dataset and detector are among the state-of-the-art pedestrian datasets and object detectors, we plan to extend the application of our methodology. However, we believe that this work is an encouraging initiation to ground partial domain specifications for MLCs instead of fully relying on their abilities to learn the concepts.

5 Related work

The stochastic nature of ML methods and the lack of recognition of the developmental standard creates many challenges in specifying requirements for MLCs. In a previous study, the authors specified performance, robustness, reusability, and interpretability as a list of desired properties for MLCs [2]. There is a large body of work in testing and verification of the robustness of MLCs, as a desired property, by the software engineering community [19, 50]. Some existing work has explored the generation and identification of adversarial examples [18, 35]. Similarly, in our previous work, we studied multiple image transformations to assess the robustness of MLCs base on the Human Visual System (HVS) [7]. There are other attempts by different communities to specify requirements for MLCs in different types of software systems by creating (a) **component-level specifications:** to define the behavior of MLCs as a whole with respect to how they address the target applications [38]. However, in such approaches, it is unclear what is the implication of the high-level specification to the downstream MLCs development tasks, such as data collection and model selection (b) **dataset specifications:** since dataset management is critical to the overall quality of systems with MLCs [24, 25]. However, studies in this area are very limited to specific domains (c) **model specifications:** which based on the particular ML algorithms, they normally define how the theoretical properties should hold during implementation [39], and (d) **development process specifications:** of MLCs to enable a consistent training result a set of predefined steps and configurations need to be carefully followed [36]. One common challenge remains as the lack of specifications for the *domain-specific concepts*. Within the safety domain, for verification purposes, it is pivotal to construct the **traceable path** to demonstrate the compliance of source code with the design specification and coding guidelines [36]. The traceable path can support building the safety case to demonstrate that the identified hazards are sufficiently mitigated. [16] also called for building the infrastructure to support traceability in automotive software when integrating a V model for data development with the standard V model for software development, what they called a W model. A recent work [3] has demonstrated the potential of traceability by maintaining and implementing the high-level software requirements through building confidence in training data. The confidence includes nine items such as that the data are sufficient, does not contain bias, and is self-consistent. However, the specific step to achieve this confidence in the training data is still open to question. For example, for the automated pedestrian collision avoidance system, what are the specific criteria of *sufficiency* for managing the dataset to recognize the concept of "pedestrian"? As we described earlier, the difficulty starts with our very limited understanding of how this concept should be defined even in the high-level specification, and how it is presented in the training data. Our work sets off to tackle the ambiguity in domain concept semantics and create a framework for specifying and validating requirements for MLCs. Our proposed approach supports identifying the gaps among the high-level specification and data instances.

6 Conclusion

In this paper, we emphasized to formally benchmark hard-to-specify domain concepts and verify their capture in a collected dataset for software components built based on machine learning algorithms. We proposed a semi-automated approach to formally specify the domain concepts in the form of a semantic web. We selected the concept *pedestrian* from the automotive domain and used a series of machine learning algorithms to automatically create a semantic web for the concept. Since the selected concept is socially-constructed, we used social platforms, such as Twitter, Wikipedia, Google News, and Google books, to construct the semantic web. Further, as a proof-of-concept, we augmented a pedestrian dataset according to a feature in our semantic web, *wheelchair*, which was captured to be closely related to the concept *pedestrian*. Our experiments showed that the accuracy of a state-of-the-art object detector, Yolo, in detecting pedestrians has been improved after this augmentation.

The problem of specifying requirements for MLCs represents a pressing and challenging area of research need. The results presented here depict the benefit of grounding hard-to-specify domain concepts as part of the requirements specification process. While encouraging, our results are preliminary, and we aim to improve on them in future work. We plan to work on the generalizability of our approach for other concepts, such as traffic and other domains. Additionally, we will evaluate the effectiveness of the automatically generated semantic webs with domain experts and with more datasets and algorithms.

Acknowledgements This work is partially funded by the US National Science Foundation Grants CCF: 2124606. This research used resources of the data, devices and interaction Laboratory (ddiLab) at Northern Illinois University.

References

1. Arthur D, Vassilvitskii S (2006) k-means++: the advantages of careful seeding. Technical report, Stanford
2. Ashmore R, Calinescu R, Paterson C (2019) Assuring the machine learning lifecycle: desiderata, methods, and challenges. arXiv:1905.04223
3. Banks A, Ashmore R (2019) Requirements assurance in machine learning. In: Workshop on artificial intelligence safety 2019 co-located with the thirty-third AAAI conference on artificial intelligence 2019 (AAAI-19)
4. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3(Jan):993–1022
5. Bond F, Foster R (2013) Linking and extending an open multilingual wordnet. In: Proceedings of the 51st annual meeting of the association for computational linguistics, vol 1. Long Papers, pp 1352–1362
6. Bossche MV, Ross P, MacLarty I, Van Nuffelen B, Pelov N (2007) Ontology driven software engineering for real life applications. In: Proceedings of the 3rd international workshop on semantic web enabled software engineering. Citeseer
7. Hu BC, Salay R, Czarnecki K, Rahimi M, Selim G, Chechik M (2020) Towards requirements specification for machine-learned perception based on human performance. In: Proceedings of the 25th international conference on requirements engineering. IEEE, Proceedings
8. Braun M, Krebs S, Flohr F, Gavrila D (2019) EuroCity persons: a novel benchmark for person detection in traffic scenes. IEEE Trans Pattern Anal Mach Intell 41(8):1844–1861
9. Burton S, Gauerhof L, Heinzemann C (2017) Making the case for safety of machine learning in highly automated driving. In: international conference on computer safety, reliability, and security. Springer, pp 5–16
10. Burton S, Gauerhof L, Sethy BB, Habli I, Hawkins R (2019) Confidence arguments for evidence of performance in machine learning for highly automated driving functions. In: International conference on computer safety, reliability, and security. Springer, pp 365–377
11. Cleland-Huang J (2015) Mining domain knowledge [requirements]. IEEE Softw 32(3):16–19
12. Dermeval D, Vilela J, Bittencourt II, Castro J, Isotani S, Brito P, Silva A (2016) Applications of ontologies in requirements engineering: a systematic review of the literature. Requir Eng 21(4):405–437
13. Dillon TS, Chang E, Wongthongtham P (2008) Ontology-based software engineering-software engineering 2.0. In: 19th Australian conference on software engineering (ASWEC 2008). IEEE, pp 13–23
14. Dollar P, Wojek C, Schiele B, Perona P (2009) Pedestrian detection: a benchmark. In: 2009 IEEE conference on computer vision and pattern recognition, pp 304–311. https://ieeexplore.ieee.org/Xplore/home.jsp
15. Elkan C (2003) Using the triangle inequality to accelerate k-means. In: Proceedings of the 20th international conference on Machine Learning (ICML-03), pp 147–153
16. Falcini F, Lami G, Costanza AM (2017) Deep learning in automotive software. IEEE Softw 34(3):56–63
17. Guo J, Gibiec M, Cleland-Huang J (2017) Tackling the term-mismatch problem in automated trace retrieval. Empir Softw Eng 22(3):1103–1142
18. Ho Y, Wookey S (2020) The human visual system and adversarial AI. arXiv:2001.01172
19. Huang X, Kwiatkowska M, Wang S, Wu M (2017) Safety verification of deep neural networks. In: CAV'17, pp 3–29
20. ISO I (2018) International organization for standardization: ISO 26262: road vehicles—functional safety. International Standard ISO/FDIS 26262
21. Kaindl H, Kramer S (2020) Towards probability-based safety verification of systems with components from machine learning. arXiv:2003.01155
22. Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002) An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans Pattern Anal Mach Intell 24(7):881–892
23. Knight JC (2002) Safety critical systems: challenges and directions. In: Proceedings of the 24th international conference on software engineering, association for computing machinery, New York, NY, USA, ICSE '02, pp 547–550. https://doi.org/10.1145/581339.581406,
24. Kohli M, Summers R, Geis J (2017) Medical image data and datasets in the era of machine learning. JDI 30(4):392–399. https://doi.org/10.1007/s10278-017-9976-3
25. Kohli MD, Summers RM, Geis JR (2017) Medical image data and datasets in the era of machine learning—whitepaper from the 2016 C-MIMI meeting dataset session. J Digit Imaging 30(4):392–399
26. Koopman P, Osyk B (2019) Safety argument considerations for public road testing of autonomous vehicles. Technical report, SAE Technical Paper
27. Koopman P, Wagner M (2016) Challenges in autonomous vehicle testing and validation. SAE Int J Transp Saf 4(1):15–24
28. Koopman P, Kane A, Black J (2019) Credible autonomy safety argumentation
29. Krishna R, Zhu Y, Groth O, Johnson J, Hata K, Kravitz J, Chen S, Kalantidis Y, Li LJ, Shamma DA et al (2017) Visual genome: connecting language and vision using crowdsourced dense image annotations. Int J Comput Vis 123(1):32–73
30. Li Y, Cleland-Huang J (2013) Ontology-based trace retrieval. In: 2013 7th international workshop on traceability in emerging forms of software engineering (TEFSE). IEEE, pp 30–36
31. Lin TY, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, Dollár P (2014) Microsoft coco: common objects in context. arXiv:1405.0312
32. MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA, vol 1, pp 281–297
33. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781
34. NYtimes (2019) Self-driving Uber car kills pedestrian in Arizona, where robots roam. https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html
35. Rozsa A, Rudd EM, Boult TE (2016) Adversarial diversity and hard positive generation. CVPRW'16, pp 410–417
36. Salay R, Czarnecki K (2018) Using machine learning safely in automotive software: an assessment and adaption of software process requirements in ISO 26262. arXiv:1808.01614
37. Salay R, Czarnecki K (2019) Improving ML safety with partial specifications. In: Romanovsky A, Troubitsyna E, Gashi I, Schoitsch E, Bitsch F (eds) Computer safety, reliability, and security. Springer International Publishing, Cham, pp 288–300

38. Seshia SA, Sadigh D (2016) Towards verified artificial intelligence. arXiv:1606.08514
39. Seshia SA, Desai A, Dreossi T, Fremont DJ, Ghosh S, Kim E, Shivakumar S, Vazquez-Chanlatte M, Yue X (2018) Formal specification for deep neural networks. In: International symposium on automated technology for verification and analysis. Springer, pp 20–34
40. Shao S, Zhao Z, Li B, Xiao T, Yu G, Zhang X, Sun J (2018) CrowdHuman: a benchmark for detecting human in a crowd. arXiv:1805.00123
41. Singh P, Lin T, Mueller ET, Lim G, Perkins T, Zhu WL (2002) Open mind common sense: knowledge acquisition from the general public. In: OTM confederated international conferences “On the move to meaningful internet systems”. Springer, pp 1223–1237
42. Speer R, Chin J, Havasi C (2017) Conceptnet 5.5: an open multilingual graph of general knowledge. In: Thirty-first AAAI conference on artificial intelligence
43. Tang K, Zhang H, Wu B, Luo W, Liu W (2019) Learning to compose dynamic tree structures for visual contexts. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6619–6628
44. Tang K, Niu Y, Huang J, Shi J, Zhang H (2020) Unbiased scene graph generation from biased training. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3716–3725. https://openaccess.thecvf.com/menu
45. Team S, Rucinski S, Slenk E, Wigell D, Manning J (2013) Software requirements specification (SRS) automated pedestrian collision avoidance (APCA1)
46. TechCrunch (2019) Tesla model x fatal crash investigation. https://techcrunch.com/story/tesla-model-x-fatal-crash-investigation/
47. Thorndike RL (1953) Who belongs in the family? Psychometrika 18(4):267–276
48. Tilbrook M, Schwitter R (2004) Controlled natural language meets the semantic web. In: Australasian language technology workshop, vol 2, p 2004
49. Verma K, Kass A (2008) Requirements analysis tool: a tool for automatically analyzing software requirements documents. In: International semantic web conference. Springer, pp 751–763
50. Xie X, Ho JWK, Murphy C, Kaiser G, Xu B, Chen TY (2011) Testing and validating machine learning classifiers by metamorphic testing. J Syst Softw 84(4):544–558. https://doi.org/10.1016/j.jss.2010.11.920
51. Xu D, Zhu Y, Choy CB, Fei-Fei L (2017) Scene graph generation by iterative message passing. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5410–5419
52. Zellers R, Yatskar M, Thomson S, Choi Y (2018) Neural motifs: scene graph parsing with global context. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5831–5840
53. Zhang S, Benenson R, Schiele B (2017) Citypersons: a diverse dataset for pedestrian detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3213–3221. https://openaccess.thecvf.com/menu
54. Zhang S, Xie Y, Wan J, Xia H, Li SZ, Guo G (2019) WiderPerson: a diverse dataset for dense pedestrian detection in the wild. arXiv:1909.12118