The Journal of Systems & Software 186 (2022) 111211

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Automatic development of requirement linking matrix based on semantic similarity for robust software development[☆]

Dnyanesh Rajpathak[a,*], Prakash M. Peranandam[b], S. Ramesh[b]

[a] Advanced Analytics Center of Excellence, Chief Data and Analytics Office, General Motors, Warren, Michigan 48092-2031, USA
[b] Global R&D, General Motors, Warren, Michigan 48092-2031, USA

ARTICLE INFO

Article history:
Received 21 October 2020
Received in revised form 11 June 2021
Accepted 25 December 2021
Available online 1 January 2022

Keywords:
Requirement engineering
Requirement linking
Semantic similarity
Automotive
Decision support

ABSTRACT

With growing complexity of modern software, it is important that the relevant textual requirements are correctly linked into a 'requirement liking matrix' during early system development stages. The resulting requirement linking matrix highlights direct and indirect interactions between different requirements, thus facilitating improved design, development, and testing of complex software systems, e.g. automotive software, electrical/electronic architectures. The sheer volume of textual requirements collected in real-life coupled with data noises makes the task of automatic requirement linking a non-trivial exercise. In this paper, we propose a novel semantic similarity model for automatically linking different requirements to organize them into a requirement linking matrix. The model computes the similarity in terms of *term-to-term*, *tuple-to-tuple*, and *text-to-text* scores. The scores are ranked to determine whether the links are having "High", "Low", or "No" relationship with each other in a requirement linking matrix. The model is deployed as a prototype tool and its performance is validated by using the real-life data. We also compare our approach with the alternative approaches proposed in literature. The system achieved the average F1 score of 0.93 in correctly linking the heterogeneous requirements.

1. Introduction

In the last decades, the automotive industry has seen enormous growth with increasing number of features, exploding software content, and it is rapidly moving towards electrification and autonomous driving. There is an advent of technologies embedded in the form of new diagnostic sensors, modules, sophisticated embedded software systems (Benedittini et al., 2009), advanced driver assist systems (ADAS), wireless devices, and so on. As shown in Fig. 1, there is a heavy reliance on embedded modules (and corresponding module software) running in modern vehicles and it is a key driving factor in most breakthroughs, e.g. autonomous driving, connectivity, electrification, smart mobility. Not surprisingly, companies that develop the technologies for the connected and autonomous vehicles totaled more than $9.5 billion through the third quarter of 2018 (Apostu et al., 2005).

This paper is concerned with the development of the control software realizing the above-mentioned features. The control software is expected to be robust in the sense that there are strong demands on the functional correctness, safe operations, and fault tolerant capabilities, like absence of single point of failures. The foundation of a rigorous software development process is strong requirement engineering, which is the focus of this paper. With 100s of features implemented in software, the number of requirements to be managed are enormous and typically these requirements are long, spread across multiple documents. It is critical that all the requirements of different features are appropriately related or linked to each other. To make the problem even more complex, these requirements are captured by different stake holders in a free-flowing English language and not always by using a controlled set of vocabulary (Mich et al., 2004; Geravasi and Zowghi, 2005). Hence, it is a non-trivial exercise to build a complete, consistent, and unambiguous linking between the requirements to ensure the completeness and correctness of intended software functions.

Below, we show a typical example of a requirement collected in our domain. This requirement is related to the vehicle start function:

Keyless Push Button Start Feature:
Pre-condition: Transmission in PARK;

1. Brake pedal pressed
2. Press Start/Stop button
3. System cranks the vehicle

[☆] Editor: Daniela Damian.
[*] Corresponding author.
E-mail addresses: dnyanesh.rajpathak@gm.com (D. Rajpathak), prakash.peranandam@gm.com (P.M. Peranandam), ramesh.s@gm.com (S. Ramesh).

https://doi.org/10.1016/j.jss.2021.111211
0164-1212/© 2022 Elsevier Inc. All rights reserved.

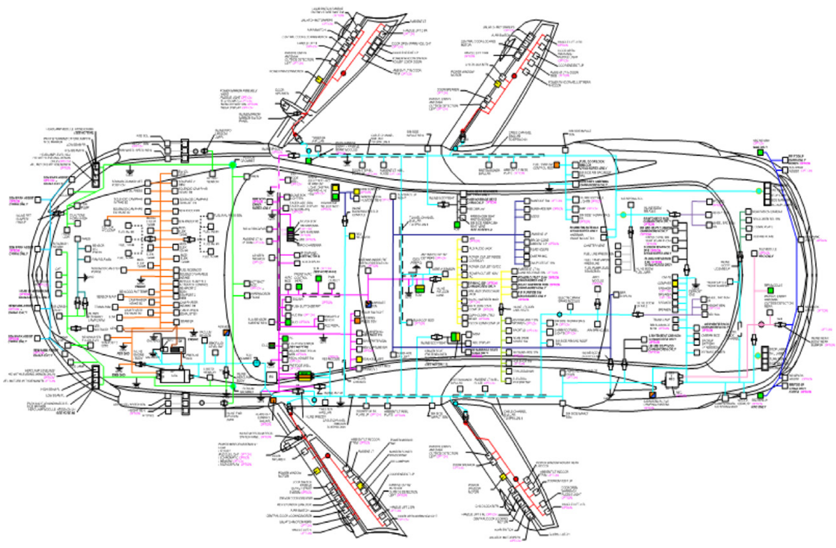


Fig. 1. Different modules and their related systems in a modern vehicle.

Post-Condition: System enables complete initialization of the required sub-systems, for example: (a) Engine initialization, (b) HMI initialization and (c) Brake initialization, in the order of "a → b → c"; and then enables driving readiness.

Remote Button Start Feature:
Pre-condition: Transmission in PARK; Vehicle system is in signal reachability state;

1. Press defined sequence of keys in the key fob in the order or through original equipment manufacturers (OEMs) mobile app from anywhere provided network connection on both ends.
2. System receives the signal for cranking.
3. System cranks the vehicle.

Post condition: System enables subset of initialization, i.e., only the required sub-systems, say, (a) Engine and (c) Brake and only with limited capabilities. System waits for the driver to enter the vehicle and trigger other procedures to complete the remaining initialization processes and then enables driving readiness within the calibrated time.

These two module specific requirements even when written in the same document can be at least tens of pages apart. Obviously, these two requirements are related and the subject matter experts (SMEs) need to maintain their relationship in the mental model such that they can be analyzed together during software development and testing. It can be imagined that the non-experts might not even realize the relationship between these two requirements. To make the problem even more complex, in some other cases, the related requirement may belong to different domains. For example, consider the safety feature "Blind Spot Monitor & Alert" requirement in the body control domain with the pre-condition: "Engine Running; and Transmission in DRIVE". A testing engineer could easily consider the pre-condition alone and perform a test. By overlooking indirect dependency relationship between two engine start feature requirements, the test engineer may miss out on considering the engine cranking sequence and the different initialization order between them. The consequence of failing to test the related requirements together may create a situation whereby the safety feature may malfunction in the remote start scenario due to the different initialization order. It is important to identify these types of indirect dependencies and interactions upfront to design and develop functionally correct software. Such indirect dependencies are difficult to extract by non-experts and given the complexity of modern vehicles there are many-to-many direct and indirect relationships exist among different features. The inherent data complexity makes it difficult or even impossible for multiple human experts to capture and maintain a mental model of relevant requirements to interpret them in conjunction.

Off-the-shelf tools, e.g. IBM DOORS©[1] provides a preliminary support for linking requirements by allowing software developers to link the requirements manually. However, manual linking has a number of shortcomings when used in real life: 1. The developers need to be thoroughly aware of the different requirements related to their domain, which is not practical due to the scale of data, 2. Since the requirements are spread across multiple pages in different requirement documents it becomes difficult to create a complete mental map of related requirements to maintain linking consistency, 3. The language used to describe requirements is non-standard, leading to *semantic ambiguity*. For example, consider the two snippets of requirements "check engine light ON" vs "Program Control Module with the fault code P0300". While these two snippets refer to the same system specific requirements, they do not have any overlapping terminology with each other. Without having a deep understanding of domain it is not possible to link such requirements as the related requirements, and 4. The requirement documents typically consists of abbreviations and it is important to disambiguate these abbreviations by using the context in which they are mentioned. It helps to avoid duplicate entries and unambiguous linking of requirements in a requirement linking matrix. For example, consider the error code "P0452: Evaporative Control System Pressure Sensor Low Input in **TPS**". It is critical to correctly disambiguate the abbreviation TPS to *tank pressure sensor* because based on the context in which it is reported the evaporative emission control system pressure sensor low input has resulted in setting the error code P0452. Given these challenges, there is an urgent need to automate the linking of heterogeneous requirements into an unambiguous, formalization of a requirement matrix by handling industrial scale data. Ultimately, our aim is to avoid incomplete requirement analysis to help design software with either limited or no erroneous behavior (Wong et al., 2010).

The ISO 26262 V-cycle Development Process sets up a framework that ties each type of testing to a corresponding design or requirement document, but it presents challenges related to

1 https://www.ibm.com/products/requirements-management.

new designs. During the initial stages of system development, it is critical to construct requirement documents as accurate as possible to capture functional operation of intended software as it is more expensive to fix an error during the later stages of development (Kotonya and Sommerville, 1998). The traceability matrix (Ramesh et al., 1997; Zowghi and Gervasi, 2003) correlates different requirements to determine their completeness relationships. Firstly, it provides system designers with a highly useful tool that provides a complete view of a system. Secondly, a requirement matrix demonstrates how a system meets user needs in terms of the requirements that are linked with each other during design, implementation, and validation (Ramesh et al., 1997; Palmer, 1997; Zowghi and Gervasi, 2003). And thirdly, it shows how the requirements of different systems are related to each other in an unambiguous fashion facilitating seamless requirement change management (Pohl, 1997). Therefore, it is evident that the interrelated requirements, that is, the ones specifying functional sub-features or the ones referring to a common part, must be checked for their consistency. Trivially, the interrelated requirements tend to have a high degree of inconsistency issues and which can unfold into the completeness and functional correctness issues.

To overcome different bottlenecks mentioned in the previous two paragraphs, in our proposal a semantic similarity model is proposed for automatically computing a semantic similarity between different textual requirements. Any two requirements with a similarity score above a specific threshold (computed empirically) are linked with each other. Finally, the linked requirements are represented in a consistent traceability matrix to improve the consistency, completeness of requirements that in turn improves the correctness. In the literature, several approaches are discussed to extract the linking between requirements in a formalized way (Egyed, 2001; Kozlenkov and Zisman, 2002; Borg et al., 2003; Nentwich et al., 2003; Gnesi et al., 2005; Lormans and Deursen, 2005; Egyed, 2006; Kamalrudin et al., 2010; Port et al., 2011; Sultanov et al., 2011; Noack, 2013). These approaches uses several independent techniques, such as traceability, semantics analysis, (semi-) formal specifications, and heuristic algorithms based on swarm technique along with the techniques from information retrieval, such as Vector Space Model, Latent Semantic Indexing, probabilistic inference network, Latent Dirichlet allocation, Jaccard index, and Cosine similarity.

Our approach extends the existing state-of-the-art of requirement linking and traceability techniques, whereby we formalize the domain knowledge in a domain ontology. This domain model allows us to tag and annotate critical technical terms mentioned in the requirement documents precisely. By doing so, we only consider relevant technical phrases while computing *term-to-term*, *tuple-to-tuple* (to handle multi-term phrases), and finally *document-to-document* semantic similarity between any two requirements. It also helps to reduce the noise commonly observed in the text data (e.g. stop words or non-technical phrases). We make the following key contributions through our work:

1. Our approach facilitates automatic discovery of not only explicit, but also implicit relations between unstructured textual requirements even when they either share little or no information at a surface level. To handle this, our model collects the context information present both at a document level as well as at the corpus level, which is used to calculate the semantic score.
2. Unlike existing approaches (Gnesi et al., 2005; Lormans and Deursen, 2005; Sultanov et al., 2011) in which the number of topics from the requirement documents are determined based on some heuristics or by the human expertise, no such constraints are imposed in our model. Instead, the domain ontology is used to identify critical technical phrases reported in the requirement documents and they are used to compute semantic similarity. Hence, in a new requirement document if the new terms are reported our domain ontology can be augmented in time for their timely identification to ensure complete coverage.
3. Our similarity model extends the classical models, e.g. the PMI-IR model (Wu and Palmer, 1994; Turney, 2002, 2006), in which the similarity between the requirement documents is calculated in terms of the three measures. First, the critical unigram terms from two requirements are used to compute *term-to-term semantic score*, $\mathrm{sim}(\mathrm{Term}_i, \mathrm{Term}_j)$, then the multi-gram phrases are used to compute *tuple-to-tuple semantic score*, $\mathrm{sim}(\mathrm{Tuple}_i, \mathrm{Tuple}_j)$, and finally the earlier two semantic similarity scores are uniquely combined into final *document-to-document semantic score*, $\mathrm{sim}(R_i, R_j)$. Through our experiments, we have shown that it helps to mitigate the key issues, such as inconsistency, incompleteness, ambiguity, and complexity. Moreover, since the linked requirements are ranked into having “High”, “Low”, and “No” relation with each other it improves the practical applicability of the requirement linking matrix. The end users can quickly focus their attention to those requirements having “High” similarity with each other.
4. From the engineering perspective, limited efforts are invested in the literature to develop a working requirement linking matrix in the automotive domain. Through our work we bridge this gap that helps to mitigate the problem of transforming unstructured requirements into a formal and structured matrix of manageable size. Since a formal requirement traceability matrix reduces the size of overwhelming data, it helps to reduce the time required to analyze heterogeneous requirements for identifying the total number of conflicting requirements.

The rest of the paper is outlined as follows: in Section 2, we present the relevant state of the art. In Section 3, we formalize our problem statement and then present different steps involved in our approach. In Section 4, first we describe the domain ontology (Section 4.1), which is used to annotate the key terms reported in requirement documents and then in Section 4.2 we describe different data cleaning steps to reduce the noise. In Section 5, we discuss in detail our model and show how the semantic similarity between any two requirement documents is computed to formalize in a structured requirement linking matrix. In Section 6, we evaluate the performance of our approach through a series of experiments. More importantly, we evaluate the correctness and accuracy of a requirement linking matrix and discuss key advantages of automatic requirement linking matrix proposal. Finally, in Section 7 we conclude our paper by highlighting the main findings.

2. Literature review

The real-life requirement documents consist of several interlinked requirements making manual modeling or their systematic structuring a tedious task for consistency and completeness checking (Ramesh et al., 1997; Zowghi and Gervasi, 2003). To overcome this limitation several techniques, such as traceability, semantics analysis, formal specifications, semi-formal specifications, and heuristic algorithms are proposed firstly to extract and secondly to link different requirements in a formal way to achieve high consistency (Egyed, 2001; Kozlenkov and Zisman, 2002; Borg et al., 2003; Nentwich et al., 2003; Gnesi et al., 2005; Lormans and Deursen, 2005; Egyed, 2006; Kamalrudin et al., 2010; Port et al., 2011; Sultanov et al., 2011; Noack, 2013; Wang et al., 2018).

In Port et al. (2011), the text mining is employed to build a traceability matrix. Many-to-many relations between the functional and non-functional requirements are organized to determine their correctness and completeness. In Lormans and Deursen (2005), the similarity (i.e. the requirements trace to each other) and dissimilarity (i.e. the requirements do not trace to each other) scores between the requirements are computed. Here, Latent Semantic Indexing (LSI) with Cosine Similarity is used to measure the similarity between different requirements. On the similar lines, in Gnesi et al. (2005), the notion of Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is exploited and they have shown promising results. The LSA model is based on Singular Value Decomposition to derive the latent semantic structure into a term-by-document matrix. The model identified the links between requirements and design along with the links between requirements and test. The model performance has been tested on the datasets of different sizes. The highest precision of 0.69 reported among all the case studies with the highest recall score of 0.89. In our domain, we observe that a dense representation of LSA makes it hard to index based on an individual dimension and it resulted into limited success. Generally, the latent topic dimension cannot be chosen to an arbitrary number, but it depends on the rank of a matrix. A key limitation of this approach is that the model is not humanly readable, and its debugging is only possible through finding the similar words for each word in a latent space. Moreover, determining the number of topics is based on the heuristics and it requires deep understanding of a domain. Finally, the Cosine Similarity measure in their approach assumes that the phrases in any two requirements have a certain degree of overlap with each other in their surface representation. In our domain, it cannot be guaranteed to have a such a type of overlap between two requirements as we do not provide any standard vocabulary to generate requirements. Hence, our semantic similarity model must address the cases when there is an either limited or no overlap of terms in two requirements. In both the cases, the model needs to compute the similarity between requirements with high accuracy.

A traceability approach with a visualization capability is proposed (Kamalrudin et al., 2010) in a light-weight automated tracing tool to perform the inconsistency checking between textual requirements. In Egyed (2001), the active and passive consistency check is covered, whereby the requirements are translated into the unified modeling language (UML) diagrams for their formal interpretations (Egyed, 2006; Nentwich et al., 2003) for their comparison. The consistency checking of design specifications that are written in a natural language is performed by using a knowledge-based approach. The design specifications are represented as the UML models with their functional requirements. The goals are represented in terms of the axiomatic statements.

The approach proposed by Kozlenkov and Zisman (2002) identifies inconsistencies observed in the design models and they are used to determine the consistency rules. The rules can be modified without having to do special annotations, but the rules are treated as a black box. The behavior of the rules can only be observed during their evaluation.

In Farfeleder et al. (2011), the domain ontologies are used to formalize the requirements captured in a natural language to ensure their consistency, completeness, and correctness. To overcome an error-prone manual process of requirement formalization, a tool named DODT is proposed. It transforms textual requirements into semi-formal requirements by using the domain ontologies. DODT also facilitate categorization of the requirements into their relevant clusters based on user defined sub-sets, manage requirement conflicts, and perform requirement tracing.

In the last decade or so, different techniques in information retrieval are employed to requirement linking and testing. In specific, a three-layer model is proposed by Noack (2013) in which the DOORS modules are used. In the first layer, the links between test cases are traced, in the second layer the elaborate filter mechanisms are applied, and finally in the case-based reasoning is used to determine the cases in which the test-link can be retrieved for their reuse in the third layer. In Borg et al. (2014), a probabilistic information retrieval model is employed to trace the recovery and to compute and rank scores as the probability between a requirement document and the source codes generated. Different evaluation measures are discussed to evaluate the performance of a model. In Marcus and Maletic (2003), a probabilistic LSI model is proposed to automatically identify the traceability links from system documentation to program source code. Finally, different evaluation measures are discussed in Parvathy et al. (2008) and Dekhtyar et al. (2007) to evaluate a traceability matrix. All the approaches discussed here are instrumental in identifying inter-document relations, document dependency, and the document consolidation with the retrieval of requirements associated with the artifacts. Generally understood, the terms in the requirement documents are retrieved and they are used to represent the documents either as a bag-of-words or as a vector in a high dimensional space, or as the stochastic variables. Different term weighting measures coupled with the semantic similarity measures, such as Jensen–Shannon divergence (Manning et al., 2008), cosine similarity (Manning et al., 2008), and Jaccard distance (Manning et al., 2008) are used to compare the similarity between requirements traceability (Abadi et al., 2008). In comparison with these approaches, in our approach a formal domain ontology is used to identify critical multi-term phrases from different requirement documents. And as it is shown in Rajpathak and Singh (2014), information extraction with the help of domain knowledge builds an enhanced association between the relevant documents when compared with the models, e.g. LDA model, that constructs the β and $m \times k$ matrices. Also, in comparison with the basic semantic similarity measures reported earlier, in our approach a hierarchical semantic similarity model is proposed to compute the similarity between any two requirements. In our model, a semantic similarity is computed in terms of *term-to-term semantic score*, $\mathrm{sim}(\mathrm{Term}_i, \mathrm{Term}_j)$, *tuple-to-tuple semantic score*, $\mathrm{sim}(\mathrm{Tuple}_i, \mathrm{Tuple}_j)$, and these two scores are aggregated into *document-to-document semantic score*, $\mathrm{sim}(\mathrm{R}_i, \mathrm{R}_j)$ to compute the semantic similarity between two requirements.

In Wang et al. (2018), a novel method based on artificial neural networks (ANN) is introduced to enhance the capability of automatically resolving polysemous[2] terms. In specific, a feedforward neural network is built that leverages a term's highest-scoring coreferences in different requirements. It helps to learn if a term has the same meaning in different requirements. The cluster-pair ranking model is proposed to detect a term coreference. A term, say 't' and its highest scoring coreference in two requirements, say r_i and r_j is generated through a pooling layer and a single layer neural network is trained. In cases, where the output is false, the term-by-requirement matrix is updated by separating the column of 't' into two different columns. The information retrieval based trace link recovery approach is employed to calculate the similarities of requirements pairs by using updated term-by-requirement matrix.

Finally, in Mahmoud and Niu (2015) a potential benefit of utilizing natural language semantics have been investigated in automated traceability link retrieval. The authors have investigated wide spectrum of information retrieval methods based on semantic information. Some of the key methods include, latent semantic indexing (Deerwester et al., 1990), vector space model (Rosario, 2000), latent Dirichlet allocation (Blei et al., 2003), vector space

[2] Polysemy refers to the coexistence of multiple meanings for a term reported in different requirement documents.

model with thesaurus support (Huffman-Hayes et al., 2003), normalized Google distance (Cilibrasi and Vitanyi, 2007), explicit semantic analysis (Gabrilovich and Markovitch, 2007), and part-of-speech enabled vector space model (Capobianco et al., 2013). These methods are selected due to their suitability in capturing and presenting requirements traceability links in software systems.

We also review relevant techniques proposed in the literature related to semantic similarity for comparing text documents. These techniques are broadly classified into — latent semantic indexing (LSI) (Deerwester et al. 1999), point-wise mutual information (PMI) (Turney, 2001), corpus based semantic representation (Riloff and Shepherd, 1997; Mihalcea et al., 2006), and ontology-structure based measures (Rada et al., 1989; Roark and Charniak, 1998; Leacock and Chodorow, 1998; Li et al., 2003). In LSI (Deerwester et al. 1999), the documents are automatically indexed and retrieved by analyzing the relationships between documents based on the terms contained in them. A term-document matrix represents the occurrences of words in text and the singular value decomposition reduces the number of columns by maintaining a semantic structure of rows. In LSI, only single term words are used during the construction of a term-document matrix, while in our approach both the single-term and multi-term words are used as the key features to compute a document similarity. In our approach, we also consider document and corpus level context information to compute *term-to-term*, *tuple-to-tuple*, and *document-to-document* semantic similarity scores.

In the PMI model (Turney, 2001), the notion of co-occurrence between any two text phrases, say $phrase_i$ and $phrase_j$ is computed in terms of the product of their co-occurrence probability. The two measures, such as *attributional similarity* and *relational similarity* (Wu and Palmer, 1994; Turney, 2002) are used to determine the similarity between two phrases. In attributional similarity, $phrase_i$ and $phrase_j$ are said to be similar with each other if they share a common set of properties. In relational similarity, the similarity between two-word pairs associated with these phrases, say $phrase_i = (\mathrm{w_i}, \mathrm{w_j})$ and $phrase_j = (\mathrm{w_k}, \mathrm{w_m})$ is considered based on a degree of correspondence between $phrase_i$ and $phrase_j$. In comparison with (Turney, 2001) in which the similarity is calculated by using single word phrases, our model extends it by constructing multi-term phrases. Our model also successfully handles data sparseness while calculating the similarity of high-order n-grams, e.g. three-four-five grams.

The main idea behind calculating the semantic similarity by using the corpus-based model (Riloff and Shepherd, 1997; Mihalcea et al., 2006) is to characterize the meaning of linguistic expressions in terms of Distributional Semantic Model (DSM). The DSM model relies on the distributional hypothesis while calculating similarity in terms of the *attributional similarity* and the *relational similarity* (Wu and Palmer, 1994; Turney, 2002). The distributional data is represented by unstructured co-occurring relations between an element and a context, where the context is defined as the lexical collocates with a certain distance from a word. The main limitation of this model is that it ignores the linguistic structure while computing the co-occurrences between words. This leads to building incorrect association between words simply because they appear close to each other in a word window of specific size. For example, in the text snippet "Battery replaced for dead vehicle", the word "replaced" gets wrongly associated with 'vehicle' simply because it appears in a word window. In our model, the domain ontology along with probability of co-occurrence helps us to discard irrelevant term associations.

In the ontology-structure based measures, the similarity between different phrases is calculated based on the concept and relation instances that are formalized in the ontology. In such models, if the words with same semantic label are found in a same ontology hierarchy then they are said to be similar with each other. In Rada et al. (1989), the bootstrapping is used to construct semantic lexicons by using the context information with one noun to the left and right of the head nouns. Roark and Charniak (1998) extended the approach by looking for the words co-occurring in a syntactic formation. Finally, the major shift in the research is proposed by Biemann et al. (2004), in which the lexical-semantic resources are built by using sentence-based co-occurrence statistics. In Mohammad and Hirst (2012), several semantic distance distributional measures are reviewed, such as WordNet's[3] semantic network-based approach (i.e. edge counting), lowest common subsumer, and dictionary-based approaches to compare the target word with those of its context. In Budanitsky and Hirst (2006), different measures, e.g. dictionary-based approach, Roget-structured thesauri, WordNet and other semantic networks, taxonomic path length, scaling the network are evaluated based on semantic distance. In Patwardhan et al. (2003), the Adapted Lesk Algorithm (Banerjee and Pedersen, 2002) is extended in which the ontologies are used to generate the word pairs. In Harispe et al. (2014), an extensive study of different ontology-based semantic similarity measures, such as edge-based approaches, node-based approaches (e.g. feature based strategies, information theory-based strategies), and hybrid approaches are conducted. A common unifying framework for ontology-based semantic similarity is also discussed.

In comparison with these approaches, in our approach we initially make use of the domain ontology to tag key technical terms and then document and corpus level context information is collected. The vectors are constructed using context information to compute a hierarchical semantic similarity scores, such as term-to-term, tuple-to-tuple, and document-to-document. Finally, in the existing literature there is limited work that combines semantic measures between requirement documents to construct an operational requirement linking matrix. Through our work we bridge this gap by showing how an operational requirement linking matrix can be built to compare industrial scale requirement data.

3. Problem statement and the methodology

We frame the problem of requirement linking in terms of computing similarity between any two requirements specified either within or across requirement documents. Formally, we define the task of semantic similarity in term of the function shown in Eq. (1):

$$\sigma_k = R_k \times R_k \rightarrow \mathbb{R}^+ \tag{1}$$

where,

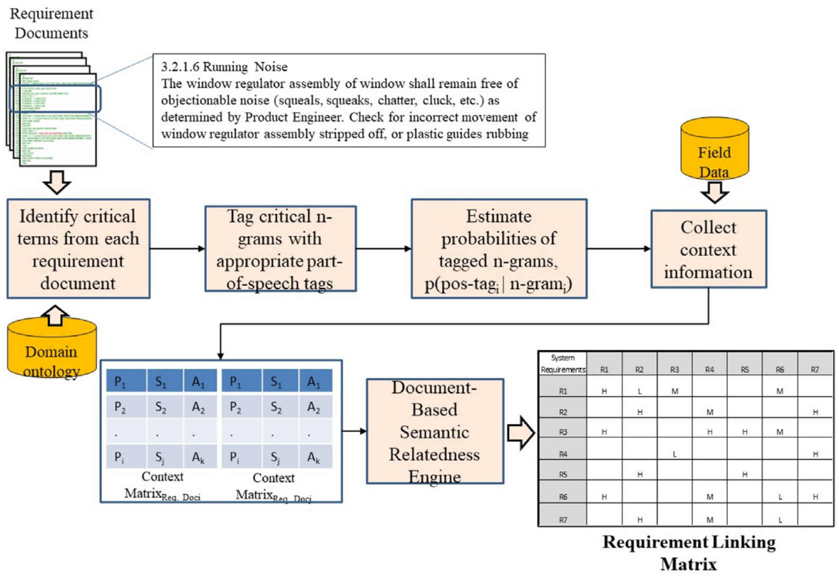
R_k is the set of elements of type k and $k \in \mathbb{K}$ and $\mathbb{K}$ consisting of different elements, which can be compared with respect to their semantics and $\mathbb{K} =$ words, phrases, context, parts of speech, etc.

In line with the definition given in Eq. (1), we also consider two important measures, viz Distance and Similarity, which enable similarity measure between requirements possible.

Definition. A *Distance*, $\mathcal{D}$, is a function that represents the domain of elements, i.e. requirements to compare, $\boldsymbol{dist} : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{V}$, where $\mathcal{V}$ is a distance on $\mathcal{D} if \forall_{Req_i, Req_j} \in \mathcal{D}$. The *Distance*, $\mathcal{D}$ has the following three properties:

- Non-negative, $dist(Req_i, Req_j) \preceq 0_{\mathcal{V}}$
- Symmetric, $dist(Req_i, Req_j) = dist(Req_j, Req_i)$

[3] https://wordnet.princeton.edu/download.


Requirement Documents
Identify critical terms from each requirement document
Domain ontology
Tag critical n-grams with appropriate part-of-speech tags
Estimate probabilities of tagged n-grams, p(pos-tag$_i$ | n-gram$_i$)
Field Data
Collect context information
Context Matrix$_{Req\_Doci}$
Context Matrix$_{Req\_Docj}$
Document-Based Semantic Relatedness Engine
Requirement Linking Matrix

3.2.1.6 Running Noise
The window regulator assembly of window shall remain free of objectionable noise (squeals, squeaks, chatter, cluck, etc.) as determined by Product Engineer. Check for incorrect movement of window regulator assembly stripped off, or plastic guides rubbing

P_1	S_1	A_1	P_1	S_1	A_1
P_2	S_2	A_2	P_2	S_2	A_2
.	.	.	.	.	.
P_i	S_j	A_k	P_i	S_j	A_k

System Requirements	R1	R2	R3	R4	R5	R6	R7
R1	H	L	M			M	
R2		H		M			H
R3	H			H	H	M	
R4			L				H
R5		H			H		
R6	H			M		L	H
R7		H		M		L	

Fig. 2. Different steps involved in the hierarchical semantic similarity model.

- Reflexive, $dist(Req_i, Req_i) = 0_{\mathcal{V}}$ and $\forall_{Req_j} \in \mathcal{D} \wedge Req_j \neq Req_i : dist(Req_i, Req_i) \prec dist(Req_i, Req_j)$

Definition. A *Similarity*, is a function, ***sim***: $\mathcal{D} \times \mathcal{D} \rightarrow \mathcal{V}$, where $\mathcal{V}$ is the distance computed based on $\mathcal{D}$, if $\forall_{Req_i,Req_j} \in \mathcal{D}$. The function ***sim*** obeys the same properties related to $\mathcal{D}$, i.e. Non-negative, Symmetric, and Reflexive. We rank the ***sim*** into the following three categories: **1.** $(sim\left(Req_i, Req_j\right) = High_V)$, and $\forall_{Req_i,Req_j} \in \mathcal{D} \wedge Req_i \neq Req_j$ and $sim\left(Req_i, Req_j\right) > \Theta_1$, **2.** $(sim\left(Req_i, Req_j\right) = Low_V)$, and $\forall_{Req_i,Req_j} \in \mathcal{D} \wedge Req_i \neq Req_j$ and $sim\left(Req_i, Req_j\right) > \Theta_2$, and **3.** $(sim\left(Req_i, Req_j\right) = No_V)$, and $\forall_{Req_i,Req_j} \in \mathcal{D} \wedge Req_i \neq Req_j$ and $sim\left(Req_i, Req_j\right) > \Theta_3$, where Θ_1 (greater than 0.92), Θ_2 (less than 0.92 but greater than or equal to 0.65), and Θ_3 (less than 0.65). The threshold values of Θ_1, Θ_2, and Θ_3 are determined purely empirically. Finally, the three similarity events, i.e. $(sim\left(Req_i, Req_j\right) = High_V) \cap (sim\left(Req_i, Req_j\right) = Low_V) \cap (sim\left(Req_i, Req_j\right) = No_V) = \phi$.

In Fig. 2, we show different steps involved in our approach to construct a requirement linking matrix from the raw requirement documents.

In our model the process starts with a set of requirement documents collected from different sources. During the step, *'Identify critical terms from each requirement document'* we make use of the domain ontology to identify critical technical terms (also referred to as the n-grams) specified in each requirement document. In particular, the concept instances in the domain specific ontology[4] are used to match the critical technical terms. For instance, the domain ontology class, 'Module' may have instances, such as 'engine control module', 'fuel tank control module', etc. and such concept instances are used to tag the terms in the requirement documents. Having identified the technical terms, in the step *'Tag critical n-grams with appropriate part-of-speech tags'* a normalized frequency of each technical term is calculated at a corpus level. The technical terms with high normalized frequency are kept as the critical ones and the part-of-speech tag is assigned to them, e.g. engine/NN control/NN module/NN has/VBZ internal/JJ short/JJ.[5] In the step, *'Estimate probability of tagged n-grams, p (pos-tag$_i$* | n-gram$_i$*)'*, the probability of n-grams, say n-gram$_i$ having a specific part-of-speech tag, say pos-tag$_i$ in a requirement document is estimated. It helps to establish true association between a n-gram$_i$ and its pos-tag$_i$. All the critical n-grams are used, and the context information associated with them is collected in the step, *'Collect context information'*. The context information is collected from the field verbatim collected in the *'Field Data'* database. Our semantic similarity model consists in the step *'Document Based Semantic Similarity Engine'*, which is used to compare any two requirements and formalize them into a requirement linking model. The context information is calculated first at a document level and then at a corpus level. Ours is a hierarchical semantic similarity model. In this model, initially the similarity is calculated between the terms identified from any two requirement documents, referred to as *term-to-term semantic score*, say sim(Term$_i$, Term$_j$). Having calculated the similarity between unigram terms, next the similarity between n-gram phrases (i.e. the ones consisting of multiple terms) is computed and it is referred to as the *tuple-to-tuple semantic score*, say sim(Tuple$_i$, Tuple$_j$). Finally, the *term-to-term* and *tuple-to-tuple* semantic similarity scores are combined to calculate document level semantic similarity score, say *document-to-document semantic score*, say sim(R_i, R_j). The document level semantic similarity score determines whether any two requirements can be linked to each other. The semantic similarity score is used to organize their linking into "High", "Low", or having "No" link with each other. In the step, 'Requirement Linking Matrix' a requirement linking matrix is generated as the output in which the rows represent different requirements and the columns indicate whether the requirements are linked to each other.

4. Document based semantic similarity model to generate requirement linking matrix

In this section, we discuss different components included in our model that are used to engineer a requirement linking matrix in detail. First, we discuss the domain ontology, which is used to tag the critical terms reported in the data. Then, the data-preprocessing is discussed in detail that is used to clean noises observed in the data. Finally, the hierarchical semantic similarity model is discussed in detail.

[4] The domain specific ontology is augmented periodically to ensure new terms are learned from the data and to avoid the domain ontology obsolescence.

[5] Interested readers can find list of part of speech tags here: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

4.1. Domain specific ontology

The domain ontology is constructed by learning the class instances from the corpus of requirement documents. The Resource Description Framework/Schema (RDF/S[6]) is used to formalize the extracted ontology. Through the steps given below, we describe a process to construct an ontology and more detailed information can be found in (Rajpathak, 2013).

Step 1. To train the ontology learning model, we randomly selected 95 000 requirement documents as a training dataset. The training dataset consists of different types of special characters as shown follows: (\"- -.,<\\=@!"/"#/&%>#+?:;'_- +\\s*). These special characters are removed to reduce noise in the data and ultimately reduces the unwanted data dimensions.

Step 2. Having cleaned the requirement documents, each requirement document is converted into unigrams and a POS tag is assigned to each unigram, e.g. headlamp/NNP bulb/NNP blown/NNP. To assign POS tags we employ Stanford's part-of-speech tagger (Ratnaparkhi, 1996; Toutanova and Manning, 2000). In our data, the length of the critical phrases span across one-term to five-terms and, therefore, each requirement document is converted into the n-grams of 1–5 length long n-grams, e.g. battery/NN, front/NNP headlight/NNP, engine/NN control/NN module/NN, and so on. To ensure that the most correct n-gram is assigned with a correct class label, e.g. part, symptom, only a longest n-gram is assigned a technical class label, and all its substrings are assigned non-technical label. For example, if the n-gram, say engine/NN control/NN module/NN is assigned as a technical class label, then its sub-grams, such as engine/NN, control/NN, module/NN, engine/NN control/NN, and control/NN module/NN are assigned with the non-technical class label. It is important to remember that in one document a specific n-gram, say engine/NN, may be marked as the non-technical, but in another document the same n-gram may be assigned with the technical class label.

Step 3. Given the scale of real-world data, it is important to identify only the critical n-grams from the data (the most relevant in the technology space). The criticality of each n-gram is determined by estimating its probability. Each n-gram is treated as a query, say q over a set of documents in a corpus, say d. While estimating the probability we assume that all the n-gram queries, q are generated independently from the data and Eq. (2) is used to estimate the probability.

$$p\,(q|d) \approx \prod_{w\in q} p(w|d) \tag{2}$$

where,

$p(w|d)$ represents the n-gram (w) probability for document d.

The n-grams with their probability above the specific threshold of 0.87 (determined empirically) are treated as the critical ones.

Step 4. Having identified the critical n-grams, our model categorizes the POS patterns that are uniquely associated with each class label, i.e. the *part POSs*, the *symptom POSs*, the *action POSs*, and the *pre-condition POS*. The class instances in the domain ontology are used to tag part, symptom, action, and pre-condition terms in the data. A set of POS patterns uniquely associated with these class instances and their probabilities are captured such that they can be used to tag the unseen data. In cases where the POS pattern is related to two different classes, say a part and a symptom, or a symptom and a precondition, or a symptom and an action, or a symptom and a precondition, or an action and a precondition, it is important to determine the most relevant association. To brake the tie, the probability of such overlapping POS pattern, say $POS_{overlap}$ is estimated with different term types, say $p\left(\cdot_k|POS_{overlap}\right)$, where $\cdot_k$ represents overlapping term types, i.e. $Part_i$, $Symptom_i$, $Action_i$, or $Precondition_i$. Through Eqs. (3)–(5) the probability calculation of an overlapping POS pattern when associated with $Part_i$ is shown. The probability calculations of other cases, such as $\mathrm{p(Symptom_j|POS_{overlap})}$, $\mathrm{p(Action_m|POS_{overlap})}$, or $\mathrm{p(Precondition_n|POS_{overlap})}$ can be realized on the same lines.

$$p\left(Part_i|POS_{overlap}\right) = \frac{p(Part_i \cap POS_{overlap})}{p(POS_{overlap})} \tag{3}$$

$$\begin{aligned} p\left(Part_i|POS_{overlap}\right) &= p\left(Part_i|POS_{overlap}\right) p\left(Part_i\right) \\ &= p\left(Part_i|POS_{overlap}\right) p(POS_{overlap}) \end{aligned} \tag{4}$$

$$p(POS_{overlap}|Part_i) = \frac{p(Part_i \cap POS_{overlap})}{p(Part_i)} \tag{5}$$

The POS pattern with the highest probability score related to a specific class is considered as the most likely class label membership of an overlapping POS pattern.

In Xu et al. (2020), Rajpathak et al. (2020) we discuss in detail how a machine learning classifier is trained to extract and classify technical and non-technical terms in the data and assign the most specific class labels, e.g. part, symptom. The classifier allows us to automatically learn new class instances from the data for the in-time augmentation of the domain ontology. The newly extracted ontology has the following structure: $\mathrm{Req_{onto}} = (\mathrm{C_i}, \mathrm{C_{subclass}}, \mathrm{R_{ci\rightarrow cj}}, \mathrm{I_{Ci}})$. The overall class structure of the domain ontology is shown in Fig. 3.

The $\mathrm{C_i}$ represents the key concepts observed in the requirement documents, i.e. `Part`, `Symptom`, `Action`, and `Pre-Condition`. The concept `Part` is used formalize different modules, submodules, systems, subsystems, and down to the individual components involved in a vehicle architecture. The concept `Symptom` is used formalize different fault symptoms observed in the event of fault. The notion of symptom is particularly important as it provides necessary clues for taking the appropriate corrective actions. The concept `Action` is used to formalize different corrective actions performed to fix the faulty behavior of a system. Finally, the concept `Pre-Condition` is used to formalize different conditions mentioned in requirement documents to be true for a specific requirement, e.g. 'engine is running, and driver door is not open'. In our domain, a class-subclass hierarchy, say $\mathrm{C_{subclass}}$ is used to formalize generic and specific concepts. In a class-subclass hierarchy the top-level concepts represent a generic set of concepts e.g. `Symptom`, whereas the specific concepts are formalized by the subclasses, such as `observable_symptom` and `non_observable_symptom`. We further specialize concept `observable_symptom` into text symptom, e.g. leakage, vehicle stalling and the error fault code symptom, e.g. P0300. The binary relationships $R_{\mathrm{ci\rightarrow cj}}$ allows us to formalize how two concepts are associated with each other. Specifically, the relation `faultFixedBy(Part`$_\mathrm{i}$ `Symptom`$_\mathrm{j}$`)` states that there exists a part and a symptom associated with a part in the event of a fault or a failure. Finally, the instances $\mathrm{I_{Ci}}$ are used to represent domain specific atomic objects which are reported in the documents. These instances are extracted from the data (by the classifier) and they are used to instantiate different concepts in the domain ontology. For example, the concept `Symptom` can be instantiated with its instances, such as '`System Locks All Doors`', '`windowRollingUp`', and so on.

The existing version of the ontology consists of more than 25000 parts, 786 actions, 1678 symptoms, and 973 preconditions.

[6] RDFS is a World Wide Web Consortium (W3C) standard for the specification of Meta data model.

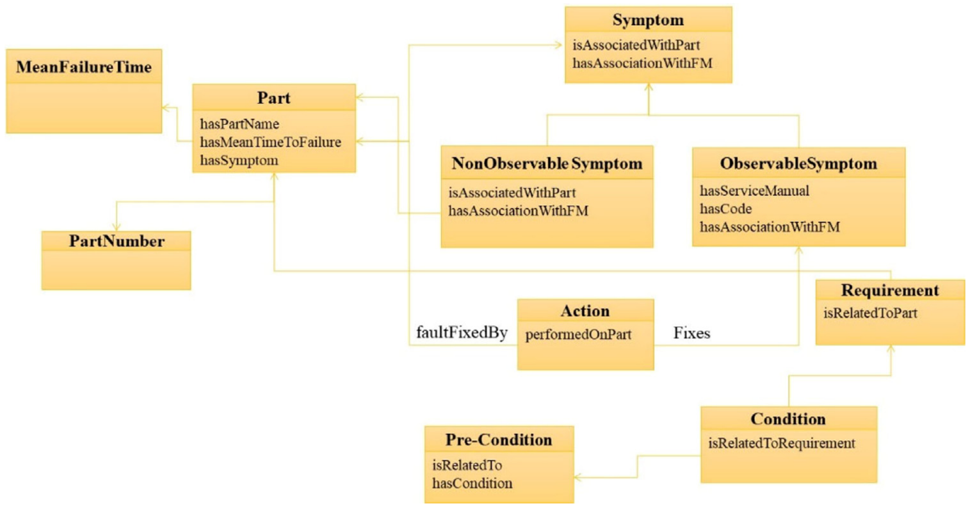


Fig. 3. The overall structure of the domain ontology showing important classes and subclasses along with relationships between two classes.

4.2. Data pre-processing

Each requirement document is preprocessed by removing the special characters observed in the data as shown follows: (\"- -.,<\\=@!"/"#/&%>#+?:;'_- +\\s*). The non-descriptive stops words, such as 'a', 'an', 'the' are also deleted as they do not add any value to the analysis. However, while deleting the stop words our model ensures that the stop words that are member of the critical terms are not deleted. It helps to maintain the original meaning of a document. The following steps are used to ensure that only those stop words that are not a member of any critical term are deleted:

Step 1. Each requirement snippet is converted into a set of n-grams of length (1, 2, 3, 4, 5), i.e. (t_i), (t_it_j), ($t_it_jt_k$), ($t_it_jt_kt_l$), and ($t_it_jt_kt_lt_m$).

Step 2. For each n-gram its support X in a document corpus, C is calculated by using Eq. (6):

$$support\,(X, C) \equiv p\,(X, C) = \frac{\#documents\ with\ X\ and\ C}{total\#\ documents\ in\ C} \tag{6}$$

Step 3. The confidence, X in each document, $r_i \in$ R is calculated by using Eq. (7):

$$confidence\,(X \to r_i) = p\,(r_i|X) = \frac{p(X, r_i)}{p(X)} = \frac{\#\ documents\ with\ X\ and\ r_i}{total\ \#\ documents\ in\ X} \tag{7}$$

The stop words that are member of the n-grams with their support and confidence value above a specific threshold, 0.90 (determined empirically) are not deleted.

The requirement documents are generated by the different stake holders. Since no curated vocabulary is provided the requirement documents consists of lean language to specify the same engineering concept at two different places e.g., 'Engine Control Module' and 'Powertrain Control Module', which refer the same engineering system. Such circular references must be disambiguated to avoid duplicate references in a requirement linking matrix. The agglomerative hierarchical clustering algorithm (Manning et al., 2008) is employed to handle the vocabulary mismatch. The steps involved are described below.

Step 1: The algorithm takes as the input two n-gram requirement phrases, say Phrase$_i$ and Phrase$_j$ and a set of requirement documents, say RD$_i$ = (rd$_1$, rd$_2$, ..., rd$_j$). Each requirement document rd$_k$ is assigned to its own cluster based on whether it reports Phrase$_i$ or Phrase$_j$. At the end of this step, the algorithm generates a set of clusters equal to the number of requirement documents, (C_{rd1}, C_{rd2}, ..., C_{rdm}) ∈ Phrase$_i$ and (C_{rd1}, C_{rd2}, ..., C_{rdn}) ∈ Phrase$_j$;

Step 2: The clusters generated in Step 1 are merged based on their similarity with each other. The clusters are merged by using the average pairwise distance (Eq. (8)) calculated between the objects that are member of two clusters, say C_{pi} = (Phrase$_1$, Phrase$_2$, ..., Phrase $_m$) and C_{pj} = (Phrase$_1$, Phrase$_2$, ..., Phrase $_n$). Based on the distance between the clusters the two most similar clusters are merged.

$$D\left(C_{p_i}, C_{p_j}\right) = \frac{1}{N_{C_{pi}} * N_{C_{pj}}} \sum_{i=1}^{N_{C_{pi}}} \sum_{j=1}^{N_{C_{pj}}} d\left(x_i, y_j\right) \tag{8}$$

$$x_i \in C_{p_i}, y_j \in C_{p_j}$$

where,

$d\left(x_i, y_j\right)$, represent the Euclidian distance between objects $x_i \in C_{pi}$ and $y_j \in C_{pj}$ from C_{pi} and C_{pj}, i.e. $d(x_i, y_j) = (x_i - y_j)^2$.

N_{Cpi} and N_{Cpj}, represent the total number of documents in C_{pi} and C_{pj} respectively.

After each iteration, the distance between remaining clusters is updated and the process is iterated until there are distinct number of clusters, C_{pi} and C_{pj};

Step 3: Having merged the clusters, the phrase similarity is computed to check if two phrases are interchangeable. The context information co-occurring with Phrase$_i$ and Phrase$_j$ is collected from the requirement documents that are member of the clusters, C_{pi} and C_{pj} and it is used to determine if the two phrases are interchangeable. To collect the context information, the position of Phrase$_i$ and Phrase$_j$ is identified in each requirement document and the word window of three words[7] is set on the either side

[7] The word window is a tunable parameter and it can be set to two, three, and five words depending on the length of each sentence. In our data, the word window of three words yield meaningful context information.

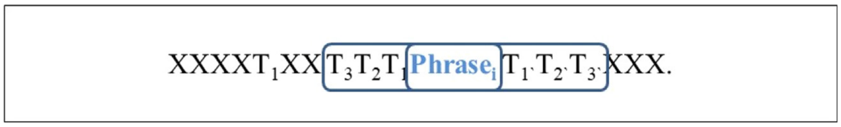


Fig. 4. The word window-based approach to collect the context information from requirement document.

of Phrase_i or Phrase_j (Fig. 4). The terms appearing within the word window are collected to generate context information, say $\text{Con}_{\text{Phrasei}} = (\text{term}_1, \text{terms}_2, \ldots, \text{term}_m)$ and $\text{Con}_{\text{Phrasej}} = (\text{term}_1, \text{terms}_2, \ldots, \text{term}_n)$.

The Jaccard similarity (Eq. (9)) is computed between the two sets of co-occurring terms, say Con $_{\text{Phrasei}}$ and Con $_{\text{Phrasej}}$ to determine the degree of overlap in the context information.

$$J\left(Con_{Phrase_i}, Con_{Phrase_j}\right) = \frac{|Con_{Phrase_i} \cap Con_{Phrase_j}|}{|Con_{Phrase_i} \cup Con_{Phrase_j}|} \tag{9}$$

The two phrases are interchangeable if the similarity score between them is greater than 0.89 (tuned empirically).

We also correct different types of noises, such as *misspelling*, *run-on-word*, and *additional white space*, which are described below.

Misspellings. To handle the misspelled word, we consider all possible corrections of a misspelled word with Levenshtein Distance equal to 1, where the Levenshtein Distance between words 'w_i' and 'w_j' is the number of deletions, insertions, or substitutions required to transform 'w_i' into 'w_j'. In cases, when a word requires only one correction, we simply replace the misspelled word with the correct word. However, when there are multiple possible corrections to a word then the one with maximum similarity score is selected as the corrected word. The similarity score is defined to be the product of its logarithm of frequency and the word2vec (Mihalcea et al., 2006) similarity between the misspelled word and its correction.

Run-on-word. To handle *run-on-words*, which represent the terms appearing as the concatenation of two words, for example, 'lighton', we split such terms into a bi-gram by inserting a white space between each pair of neighboring characters. In the case of run-on word, say 'lighton', we consider the following possible splits: 'l ighton', 'li ghton', 'lig hton', 'ligh ton', 'light on', and 'lighto n'. For a specific split if both the left-side and right-side chunk, i.e. 'light' and 'on' respectively are in the dictionary then it is regarded as the correct split. If multiple splits are possible, then for each correct split its similarity score is defined to be the maximum of word2vec similarities between the run-on word and the two chunks. The run-on word with chucks with maximum similarity score is replaced as the correct split.

Additional white space. In several cases *additional white spaces* are inserted in a word, e.g. 'actu ator'. Here, we try to remove the additional white spaces to see whether it turns the two misspelled words into a correct word if it is mentioned in the reference dictionary. For example, for two neighboring misspelled words 'actu ator' after removing the white space the new word, i.e. 'actuator' is a correct one and it is used to replace the two incorrect chunks.

5. Semantic similarity kernel to construct a requirement linking matrix

Using our semantic similarity model, the aim is to determine whether any requirements, say R_i and R_j from different requirement documents can be linked to each other based on their similarity. Each requirement document consists of several phrases (n-grams), which can be used to compare the similarity. However, not all n-grams are equally critical in the context of specified requirements. Hence, such n-grams as the ones with their criticality is above a specific threshold in documents are selected. The criticality of each n-gram is calculated in terms of the term frequency inverse document frequency (tf*idf) (Spärck, 1972) using Eq. (10). The n-grams with their tf*idf score greater than 0.75 are used for the further analysis.

$$\left(C_{n-gram}\right)_{i,j} = C_{i,j} * idf_{C_i} \tag{10}$$

$$C_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{11}$$

where,

$n_{i,j}$ in Eq. (11) is the number of co-occurrences of a given ith tuple appearing in a jth requirement document and the denominator is the sum of the number of co-occurrences of all tuples in C_i.

$$idf_{C_i} = \frac{\log |V|}{|\{v : C_i \in v\}|} \tag{12}$$

where,

Adding log to the inverse document frequency (idf_{C_i}) in Eq. (12) downweighs the effect of high frequency terms. A linear idf function may boost the document scores with high frequency terms in comparison with the terms having low frequency. However, a sublinear function (Eq. (12)) performs better when a corpus consists of terms with different frequencies.

Within our semantic similarity model, the similarity between R_i and R_j is computed in terms of the context information associated with key technical phrases, i.e. part, symptom, action and pre-condition that are identified with the help of domain ontology. The context information is collected both at a *document level* and a *corpus level*.

5.1. Context information generation at a document level

The document level context information is collected by using the critical terms $(T_1, T_2, T_3, \ldots, T_i) \in T_m$ and $(T_1, T_2, T_3, \ldots, T_j) \in T_n$ identified from R_i and R_j respectively. In our model, the context information related to each critical term consists of: 1. The co-occurring terms related to T_m and T_n that are qualified by the tf*idf vector and 2. Syntactic information, such as the parts of speech (POS) tag of each term T_m and T_n.

To generate the co-occurring terms, the algorithm iteratively selects each term from T_m and T_n and applies a word window of specific terms on the either side of T_m and T_n (as described in Fig. 4). All the co-occurring terms appearing within a word window are collected and the tf*idf value of each collected term is calculated to generate the tf*idf vectors, say $T_m = (t_{1tf*idf}, t_{2tf*idf}, \ldots, t_{itf*idf})$ and $T_n = (t_{1tf*idf}, t_{2tf*idf}, \ldots, t_{jtf*idf})$.

To generate the syntactic context information using the parts of speech (POS) tag of each term T_m and T_n, first each term in a verbatim is considered as the focal term and its POS tag is considered as the base linguistic feature. Next, a co-occurring set of terms are collected by using the word window and the POS tag of co-occurring terms is collected. Next, we calculate the

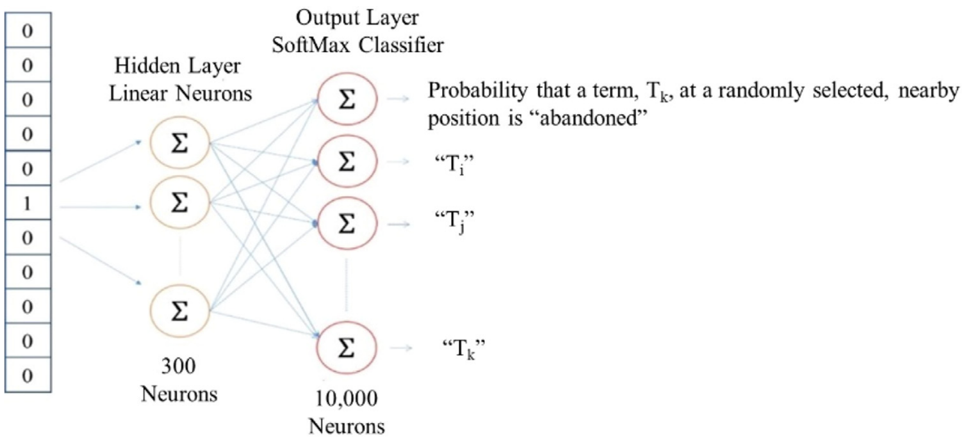


Fig. 5. A skip-gram model is trained to collect word2vec vectors for each term.

probability of $p(T_{co-occur_{pos}}|T_{m_{pos}})$ and $p(T_{co-occur_{pos}}|T_{n_{pos}})$ by using Eqs. (13) and (14).

$$p(T_{co-occur_{pos}}|T_{m_{pos}}) = \frac{p(T_{m_{pos}} \cap T_{co-occur_{pos}})}{T_{m_{pos}}} \tag{13}$$

$$p(T_{co-occur_{pos}}|T_{n_{pos}}) = \frac{p(T_{m_{pos}} \cap T_{co-occur_{pos}})}{T_{m_{pos}}} \tag{14}$$

All the probabilities are used, say $T_m = (t_{1_{co-occurpos}}, t_{2_{co-occurpos}}, \ldots, t_{i_{co-occurpos}})$ and $T_n = (t_{1_{co-occurpos}}, t_{2_{co-occurpos}}, \ldots, t_{j_{co-occurpos}})$ in the similarity calculation.

5.2. Context information generation at a corpus level

The corpus level context information is collected in terms of the word embedding. The word embedding is particularly important due to its ability to capture different types of context, such as semantic, syntactic, and relations from a corpus. The embeddings are represented in terms of the vectors of each n-gram. Different embedding models, such as word2vec, BERT[8] (Devlin et al., 2019) and Glove (Pennington et al., 2014) are proposed in the literature. Among these alternative models, the word2vec model has the following desirable properties in our domain: 1. the displacement vectors between n-grams with similar relationships are also very similar. For instance, the vector for "ECM to Powertrain" is almost the same as the vector for "PCM to Transmission", 2. In general set up, one of the common limitations of the word2vec model is that it provides smaller context with limited cooccurrence. However, in our domain it is seen as an advantage as it allows us to construct a precise context for the n-grams observed related to specific faults. By constructing a global corpus level cooccurrence context entangles the cooccurring information associated with faults (e.g. ECM — internal failure resulting in reduced propulsion vs ECM — internal short resulting in stalling). It is not desirable property in our domain since it blurs the fault isolation process, and 3. The mapping between the target n-gram to its context information implicitly embeds sub-linear relationship into the vector space of n-grams making it easy to infer the relationship between "ECM:Internal Failure as the Vehicle:Reduced Propulsion". Hence, in our approach we train the skip-gram word2vec model (Mikolov et al., 2013) to further collect context information from the corpus. The model generates a vector associated with each term, T_m and T_n and captures its context embeddings. The skip-gram model has the following interesting properties – 1. Embeddings close to each other have similar meanings and their context vector can be used within a vector operation and 2. Embeddings have semantic decompositions. The skip-gram model is trained by using ∼15.6 million documents and to limit the size of a vocabulary only the terms with frequency greater than 10,000 in a corpus are selected. A word window size of 3 is employed to collect the co-occurring context information and a hidden layer is represented by the weighting matrix with 10,000 rows (one for every n-gram in our vocabulary) and 300 columns (one for every hidden neuron), i.e. learning word vectors with 300 features. Finally, if a specific n-gram is not observed in our data or with a low frequency its vector is imputed by adding an empty vector (0, 0, …, 0). The structure of the skip-gram model is shown in Fig. 5.

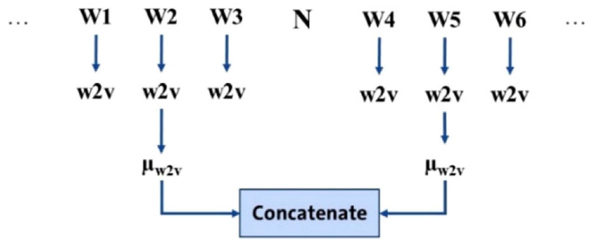


Fig. 6. The process of generating context feature for n-grams with their length > 1.

For 2,3,4-grams, they are broken into 1-grams and the average embedding of 1-grams is added as its feature vector as shows in Fig. 6.

The context information related to each R_i and R_j is a stack of document and the corpus level vectors as shown in Fig. 7.

The document level and corpus level context information associated with R_i and R_j is organized into two context matrices, $CM_{Ri} = ((T_m\ Part_i), (T_m\ Symptom_j), (T_m\ Action_k))$ and $CM_{Rj} = ((T_n\ Part_l), (T_n\ Symptom_m), (T_n\ Action_n))$ respectively. The CM_{Ri} and CM_{Rj} are used to compute the score between R_i and R_j. Initially, the model computes a *term-to-term semantic similarity* score for the phrases with only one term by using Eq. (15).

8 The authors are aware that in BERT the embeddings of a specific word are constructed dynamically depending on context in which word is mentioned. On the other hand, the embeddings are static in the word2vec model. The notion of dynamic context adaptation works well on tasks like question-answering, natural language interface, machine translation, etc. However, while working in our domain it is extremely important to preserve the context vectors comparable constructed from the data collected from one model year to another model year. The word2vec allows context vectors comparable since they are constructed from the (common) n-grams observed in the data. The dynamic embeddings of BERT provide limited insight about (common) n-grams related to a specific requirement from one model year data to another.

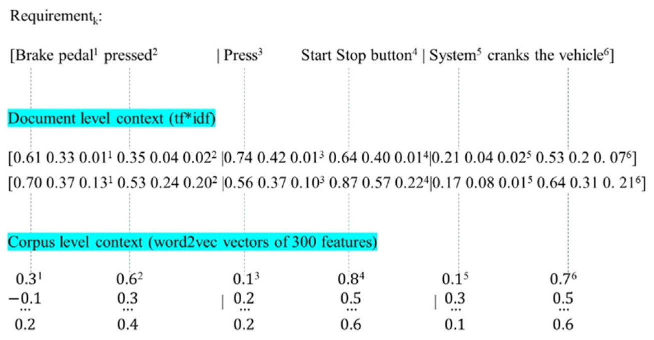


Fig. 7. A stacked document level and corpus level context vectors related to a requirement.

$$sim(Term_i, Term_j) = log_2 \left\{ 1 + \frac{hits(Term_i, Term_j)^2}{hits\,(Term_i)\,.hits(Term_j)} \right\} \tag{15}$$

where,

hits(Term$_i$), *hits*(Term$_j$) and *hits*(Term$_i$, Term$_j$) are the number of times Term$_i$ and Term$_j$ and the binary tuple(Term$_i$, Term$_j$) appear together in CM$_{Ri}$ and CM$_{Rj}$.s

Having calculated the *term-to-term similarity* score, the *tuple-to-tuple semantic similarity* score, say sim(Tuple$_i$, Tuple$_j$) is calculated for the phrases with more than one term by using Eq. (16). The *tuple-to-tuple semantic similarity* extends the standard PMI-IR model by making the following changes: (1) the square term is added in the numerator along with addition of one to the fraction. Both of these are motivated by the need to scale the measure between zero (tuples never co-occurring) and one (identical tuples or tuples always occurring together) and (2) the multi-term phrases are used while computing the score between R$_i$ and R$_j$ instead of simply using the single word phrase (as specified in the standard PMI-IR) and takes into account the context information to get a better estimate.

$$sim(Tuple_i, Tuple_j) = log_2 \left\{ 1 + \frac{hits(Tuple_i\ Tuple_j)^2}{hits\,(Tuple_i)\,.hits(Tuple_j)} \right\} \tag{16}$$

where,

hits(Tuple$_i$), *hits*(Tuple$_j$) represents the frequency of occurrence of the tuples in a corpus. The *hits*(Tuple$_i$ & Tuple$_j$) represents the number of times both Tuple$_i$ and Tuple$_j$ occurs in a corpus.

Finally, the *term-to-term* and *tuple-to-tuple* semantic similarity scores are combined to calculate the final *document-to-document* semantic similarity score between R$_i$ and R$_j$ (Eq. (17)).

$$sim\left(R_i, R_j\right) = \frac{1}{2}\left[\begin{array}{c}\left(\frac{\sum_{Tuple_i \in R_i}\left(maxsim\left(Tuple_i, R_j\right).idf\left(Tuple_i\right)\right)}{\sum_{Tuple_i \in R_i} idf\left(Tuple_i\right)}\right) \\ + \\ \left(\frac{\sum_{Tuple_j \in R_j}\left(maxsim\left(Tuple_j, R_i\right).idf\left(Tuple_j\right)\right)}{\sum_{Tuple_j \in R_j} idf\left(Tuple_j\right)}\right)\end{array}\right] \tag{17}$$

The *maxsim* $\left(Tuple_i, R_j\right)$ function in Eq. (17) is calculated by using Eq. (18). In Eq. (18) we select the maximum (max_j) semantic similarity between two tuples. In other words, the algorithm iteratively compare each Tuple $_i$ with every other tuple, $tuple_j \in R_j$. In the pairwise comparison, the pair with highest (represented as, max_j in Eq. (18)) semantic similarity is selected.

$$maxsim\left(Tuple_i, R_j\right) = max_j\left\{sim(Tuple_i, tuple_j)\right\};\ tuple_j \in R_j \tag{18}$$

The *document-to-document* semantic similarity score between R$_i$ and R$_j$ is used to determine if two requirements, R$_i$ and R$_j$ can be linked with each other. The two requirements based on their score are ranked into one of three categories, namely *high link*, *low link*, or *no link* using the following rules:

Rule 1. If the semantic similarity score between R$_i$ and R$_j$ is equal or greater than 0.70,[9] then R$_i$ and R$_j$ are said to have a high link with each other.
Rule 2. If the semantic similarity score between R$_i$ and R$_j$ is greater than 0.4, but less than 0.70, then R$_i$ and R$_j$ are said to have a low link with each other.
Rule 3. If the semantic similarity score between R$_i$ and R$_j$ is less than or equal to 0.4, then R$_i$ and R$_j$ are said to have no link with each other.

6. Experiments and discussion

Our semantic similarity model is implemented as a proof-of-concept tool on OS Type: Microsoft Windows XP Professional, Memory size: 3.5 GB, Processor: Intel[R] Core[TM]2 i7-4810MQ CPU @ 2.80 GHz. To evaluate the performance of our model we conducted a series of experiments by using the data related to “Entry Control” sub-system requirements generated during early stages of the system development. In total of 126,000 requirements associated with the different sub-systems of “Entry Control”, such as “Transmission Shifted into Park”, “Transmission Shifted Out of Park”, “Courtesy Switch Request”, “Remote Locking, Door Closed and Open”, “Relocking Without Door Open”, “Airbag Deployed”, “Remote Driver Door Only Unlocking”, “Remote All Door Unlocking”, “Key in Ignition”, “Remove Key from Ignition”, and “Continuous Press and Hold from Key Cylinder”, were used for the experiment.

6.1. Comparison with the alternative approaches

Here, we compare our model with the alternative approaches proposed in the literature, such as Vector Space Model (VSM), Latent Semantic Indexing (LSI), Pairwise semantic comparison, the Semantic Similarity model (Zesch and Gurevych, 2007) and the PMI model. We applied our model as well as the models in literature on a same set of requirements to compare their performance with each other. Table 1 shows the details of the data used for this evaluation.

All the requirements in the datasets were preprocessed to get rid of the noise. The tf*idf of each term was calculated and organized into vectors. These vectors were used to compute the cosine similarity between the two requirements to determine where they were similar with each other. In LSI, first the term-document matrix, (say *A)* was constructed and it was decomposed to find the two orthogonal matrices, **U** and **V** and a diagonal matrix, **S**. Next, the Rank-2 Approximation was implemented by keeping the first two columns of **U** and **V** and the rows of **S**. The rows of the reduced 2-dimensional space hold eigenvector values to represent the co-ordinates of each individual requirement document. Finally, the similarity between any two requirements was computed by using the cosine similarity. The similarity between two documents by using the Pairwise semantic similarity model was computed by combining the semantic similarities of the concepts involved in them. Initially, the list of documents containing the term (t) was identified and then the standard inverted index was constructed. The key tuples corresponding to the pairs were

[9] In our approach, we performed a series of experiments along with subject matter experts (SMEs) to determine the optimal threshold value.

Table 1
Data used to evaluate our model with the alternative approaches.

Dataset	Number of requirements (in thousand)	System - Subsystem
Dataset1 (d1)	42	Transmission Shifted into Park, Transmission Shifted Out of Park, Courtesy Switch Request, Remote Locking
Dataset2 (d2)	42	Door Closed and Open, Relocking Without Door Open, Airbag Deployed, Remote Driver Door Only Unlocking
Dataset3 (d3)	42	Remote All Door Unlocking, Key in Ignition, Remove Key from Ignition, Continuous Press and Hold from Key Cylinder

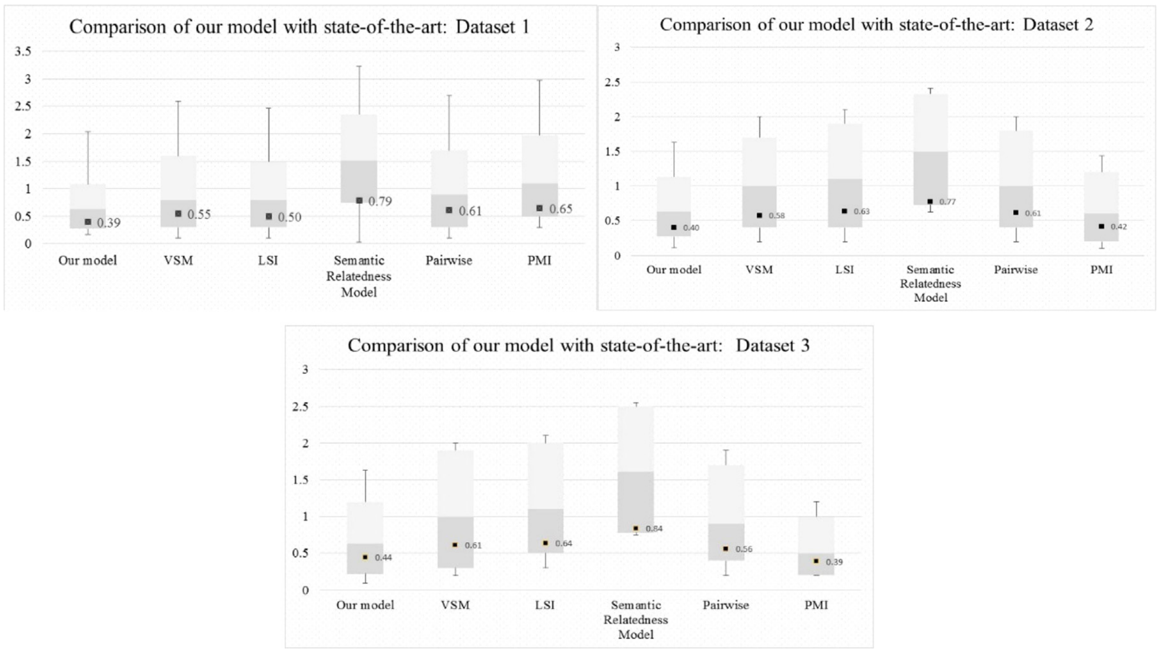


Fig. 8. Comparison between our model with VSM, LSI, semantic similarity, Pairwise, and PMI.

identified and Eq. (19) was used to compute pairwise similarity between any two requirements.

$$sim\left(d_i, d_j\right) = \sum_{t \in V} Pair_{t,d_i}.Pair_{t,d_j} \tag{19}$$

Finally, Zesch and Gurevych (2007) proposed a model in which the semantic similarity between requirement documents was computed based on the number of categories assigned to each requirement document. The best value among all pairs was selected based on the minimum for path and the maximum for information content. The results of these experiments were presented to the SMEs who in turn certified the true similarity between the requirements. In Fig. 4, we show the comparative analysis of semantic similarity calculated by our model, VSM, LSI, pairwise model, semantic relatedness model, and PMI model to see whether the requirements were related to each other.

As it can be seen in Fig. 8, the mean error of our model was lowest in all the three datasets in comparison with state-of-the-art models. The closer analysis of results revealed that the VSM, LSI, semantic relatedness, and the Pairwise models did not consider the context associated with critical terms. Hence, these models ended up missing on linking the key requirements. One of the key limitations of these models was that the key terms from two requirements documents needed an exact match to obtain a high semantic similarity score. However, in any real-life data it is difficult to control the vocabulary used by the different stake holders and it is important to identify multiple meanings of the same word by exploiting their context. For example, the following two requirements present the same engineering information, but uses different vocabulary, **R1.** *The system power mode is RUN, the engine control module operative, the brake pedal and the transmission gear not in park, ADL is enabled. The operator issues a courtesy unlock request. At least one door is opened. The brake pedal released and the system locks all doors*; **R2.** *At least one door is open and the key is in ignition. The operator issues an unlock request for the system to deactivate anti-lock*. In our model, we successfully overcome this limitation by exploiting the context information specified in the documents along with the domain ontology. As it can be seen in Fig. 8, the results of the Pairwise model were comparable, but not as good as the results achieved by our model. The main reason behind this was that the Pairwise model relied on identifying a set of documents containing common term pairs to compute the pairwise semantic matrix, but their context in the given documents was not identified. Finally, the results of the PMI model were comparable to our model in datasets 2 and 3, but our model performed better in dataset 1. In dataset 1, the PMI model considered all such terms co-occurring with critical phrases within a specific word window. Not all the terms co-occurring within a specific word window have direct association with the critical phrases and it introduced noise in the semantic similarity calculation. To ensure that correct co-occurring terms were selected for the semantic similarity calculations we used the domain ontology. The ontology helped us in selecting relevant co-occurring terms and it helped to improve the semantic similarity calculations.

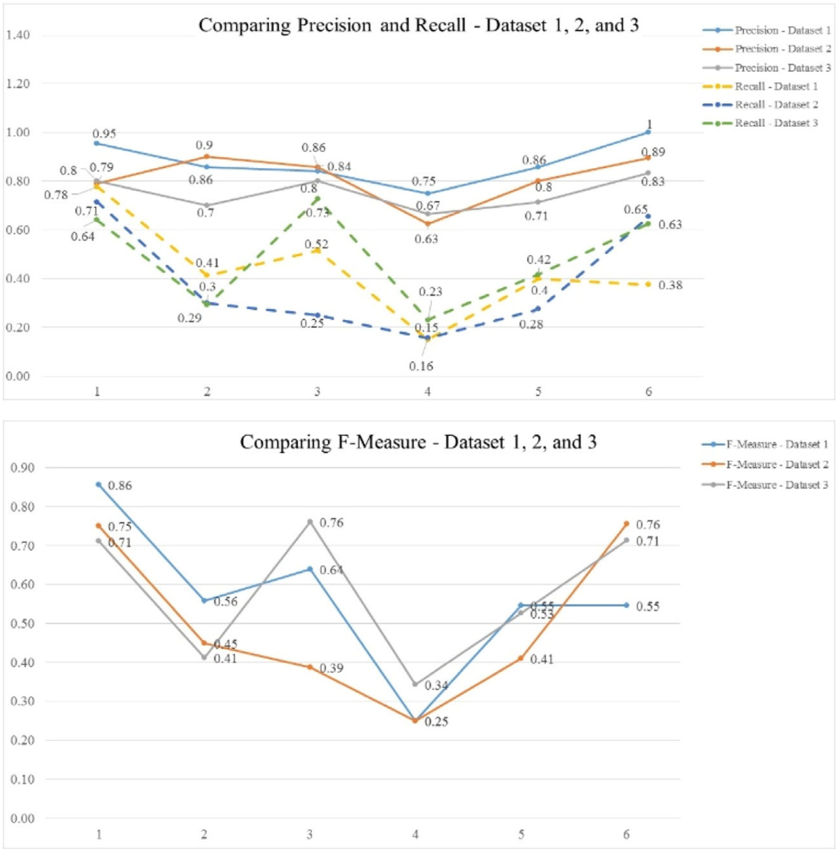


Fig. 9. Comparison of our model with other models using precision, recall, and f-measure.

We also evaluated the performance of our model with other models by using the measures, such as precision, recall, and F1-score (Fig. 9). The first dot on each line represents the performance of our model and rest of the dots sequentially represent the performance of VSM, LSI, pairwise model, semantic relatedness, and PMI respectively. As it can be seen in the datasets 1, 2, and 3 the average precision of our model was higher than other models except for the PMI model, but in all the three datasets our model outperformed all other models in terms of an average recall. In other words, our model succeeded in linking additional requirements, which other models failed to link. The results of the PMI model were comparable to our mode, but in several instances within our data when the phrases consisted of more than three terms in such cases the PMI model overestimated such low frequency terms.

6.2. Evaluation of the domain specific ontology

Here, we evaluate the performance of the domain specific ontology (Section 4.1) to see how well it tagged the critical terms specified in the data. In total, 124 requirement documents were randomly selected for the "Entry Controls" system.

In the first set of experiment, the critical terms from all 124 requirements were manually tagged by the subject matter experts (SMEs). The same set of documents were provided as the input to our key term identification algorithm, which tagged and extracted critical terms from the data. The results of our algorithm were compared with the manual annotation performed by the SMEs. The results are given in Table 2. Our algorithm identified 13 additional parts (P), 14 additional symptoms (SY), and five additional actions (A), which SMEs failed to identify during their manual annotation process.

Next, we evaluated the accuracy of the terms extraction algorithm by asking the SMEs to classify these terms as True Positives (*tp*), i.e. the terms correctly classified into their specific classes, i.e. parts, symptoms, actions, etc., True Negatives (*tn*), i.e. the tool correctly rejected the terms that does not belong its own class, False Positives (*fp*), i.e. the terms classified by the tool into a specific class that failed to match with the SME classification, and False Negatives (*fn*), i.e. the tool rejected the terms as a no-membership candidate for a specific class while the SMEs classified such terms into that class. Next, the Precision, Recall, and F1 (F-Measure) defined in Eq. (20), (21), and (22) were used.

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)} \tag{20}$$

$$Recall = \frac{True\ Positives}{(True\ Positives + False\ Negatives)} \tag{21}$$

$$F1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{22}$$

In Fig. 10, we show the summary of the ontology evaluation experiment in term of the Precision (0.72), Recall (0.75), and the F-Measure score (0.73).

6.3. Correctness of the linked requirements

In this experiment, we evaluated the correctness of the links established by our algorithm between different requirements. To develop a benchmark, we randomly selected a set of requirements and SMEs manually linked those requirements. The SMEs identified and captured in total 126 links in a matrix and classified them into "Yes/ No" categories depending on if there was a link between the requirements or not. The same set of requirements

Table 2
Summary of the results of terms identified by subject matter expert versus the ones identified by our tool.

P (SME)	P (Tool)	sy (SME)	sy (Tool)	a (SME)	a (Tool)
89	102	57	71	31	36
New parts identified by tool		New symptoms identified by tool		New actions identified by tool	
13		14		5	

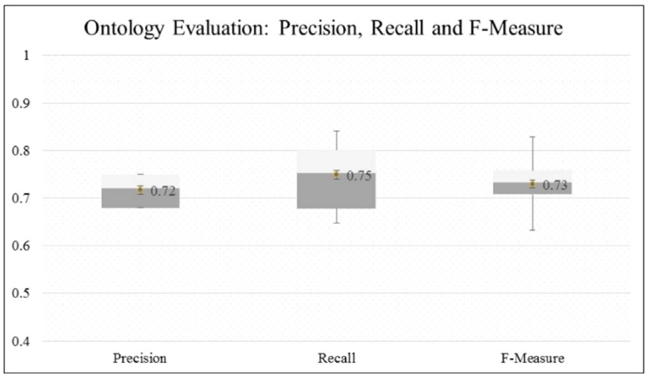


Fig. 10. The evaluation of domain ontology and term extraction algorithm in terms of precision, recall, and f-measure.

were provided as input to our algorithm to generate the automatic linking between them in a matrix. The requirements were said to be similar with each other, i.e. "Yes" if the score was greater than or equal to 0.4 and if the score was less than 0.4 then they were marked as not similar, i.e. "No". Figs. 11 and 12 shows the important portions of the result matrix by the SME and the tool respectively. The first column and the first row show the requirement unique IDs.

The analysis of the results generated by the algorithm when compared with the SME linking revealed that over 84% of the requirement linking generated by the tool were accurate to the ones identified by the SMEs. In other words, 106 links out of 126 links generated by the tool matched with the SME links, i.e. True Positive and True Negative. In the remaining ∼16% of the cases the SME's decision were "No" link, whereas the tool identified as "Yes" link, i.e. the "False Positive" linking. These false positives cases are highlighted in Figs. 11 and 12. Most importantly, there were no cases in which the tool linked a requirement as "No" link, while the SME linked it as "Yes", i.e. no "False Negative" links.

6.4. Accuracy of degree of link between requirements

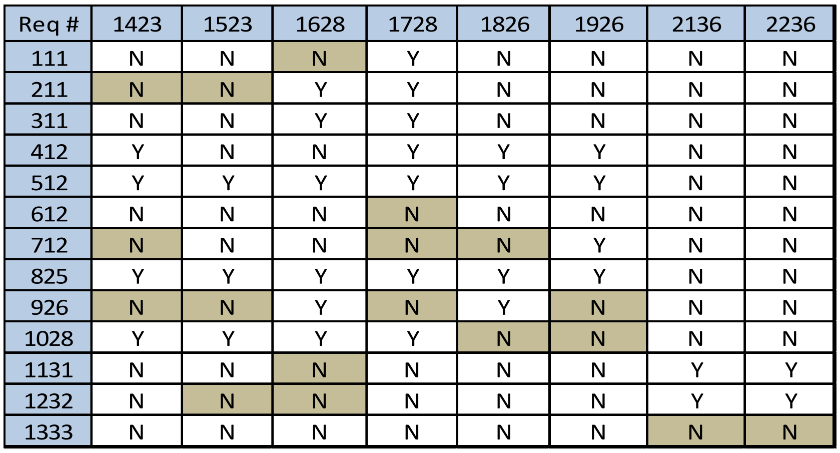
Here, we describe the validation of our approach in correctly classifying requirements with the 'High' or 'Low' categories of the link levels. Again, the categorization performed by the SMEs was considered as the benchmark. In this experiment, only two link categories were used because any two requirements can be in the sub-state space of the same functionalities or different states that may or may not be in the same functionality state space. For example, the requirements "Subsequent Press of Courtesy Switch" and "Remote Locking" deals with the "Locking" functionality and both these requirements holds the same pre-condition, except that in the former requirement mentioned that "at least one door is open" and the latter requirement mentioned that the "driver door is open". With these pre-conditions the latter requirement clearly belonged to the sub-state of the former requirement because the driver door comes as the subsystem of the door system assembly in the vehicle architecture. Whereas, the requirements "Tonneau Release - Interior Release" and "Airbag Deployment" clearly belonged to two different states as these two requirements were dealing with two independent functionalities. Trivially, highly linked requirements were studied to check their consistency issues. The thorough study was necessary because the highly linked requirements typically belonged to the same functional sub-state space, while the low linked requirements usually belonged to different states that may or may not be in the same functional space and share the same parts.

In this experimentation, we have considered 39 links out of 126 links that were linked as "Yes" and received a degree of linkage by the SME. The summary of the links identified by our approach with respect to the links identified by the SMEs is shown in Table 3. The "True Positive" cases were defined as the linking level and degree are the same as the SME degree of link, i.e., "Low" or "High". The "False Positive" cases were defined as the linking level higher compared to the SME linking, i.e., "Low" to "High". In other words, the linking produced by the tool says "High", but the SME has mentioned it as "Low". Obviously, in this experiment there were no "True Negative" and False Negative" cases because such cases were considered in the previous experiment (Experiment 2). Out of 39 linking, the tool classified 20 links as having the "Low" level of link, which matched with the links identified by the SMEs. In five cases, the tool misclassified the links as having the "High" link, when the SMEs linked them as having the "Low" link. However, for the remaining 14 linking the tool classified them as "High", which is the same as the SMEs. Table 4 shows the summary of the experiment results.

Based on the 2 × 2 table (Table 4) we have calculated the Precision, Recall, and F1-measure as defined in Eq. (20), (21), and (22) were used. On an average the Precision was 0.87, the Recall was 1 and the F1-Measure was 0.93. At the end of experiments 1 and 2, we realized that around 70% of the comparisons resulted in the no link category and about 30% of comparison resulted in linked requirements with different degree of links. Typically, in case of a manual linking where an average of 70% of the cases have no links it becomes a tedious and tiring process. Also, there is a high chance that a human reviewer may declare a linked requirement as the non-linked one due to fatigue and monotonous nature of the links. The third set of experiments was performed to validate our assumptions to reveal the necessity of the auto linking tool.

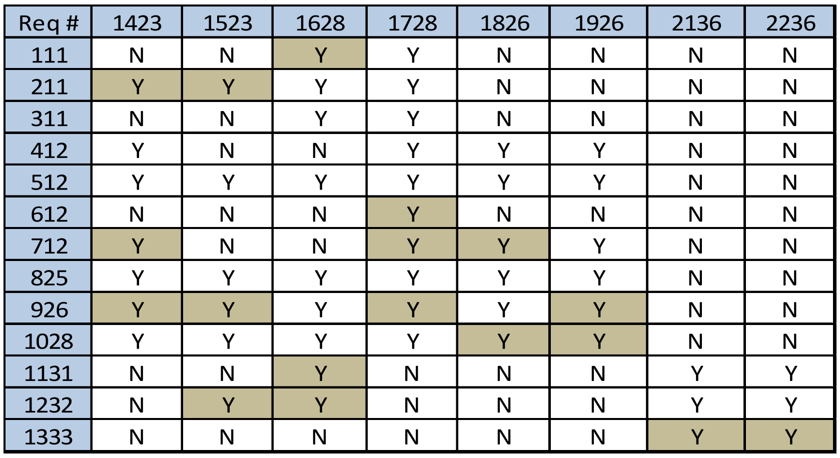
6.5. Advantages of auto linking of requirements

In this experiment, we tried two different approaches for manual linking. In the first approach, we have provided the SMEs with a requirement document and the experiment was about the study of "false positive" links with respect to the links generated by the SMEs. Initially, the SMEs developed a mental model of requirements by cross linking each new requirement with all the previously linked requirements. With approximately five requirements per page, the mental linking matrix quickly become hard to handle for the SMEs within the first few pages of the document. The difficulty increased in a linear manner to the number of requirements reviewed. In this process, the focus and the effectiveness of the review diminished at an exponential rate. There were several cases identified in which the SMEs marked



Req #	1423	1523	1628	1728	1826	1926	2136	2236
111	N	N	N	Y	N	N	N	N
211	N	N	Y	Y	N	N	N	N
311	N	N	Y	Y	N	N	N	N
412	Y	N	N	Y	Y	Y	N	N
512	Y	Y	Y	Y	Y	Y	N	N
612	N	N	N	N	N	N	N	N
712	N	N	N	N	N	Y	N	N
825	Y	Y	Y	Y	Y	Y	N	N
926	N	N	Y	N	Y	N	N	N
1028	Y	Y	Y	Y	N	N	N	N
1131	N	N	N	N	N	N	Y	Y
1232	N	N	N	N	N	N	Y	Y
1333	N	N	N	N	N	N	N	N

Fig. 11. A snapshot of the requirement linking matrix generated by the subject matter expert.



Req #	1423	1523	1628	1728	1826	1926	2136	2236
111	N	N	Y	Y	N	N	N	N
211	Y	Y	Y	Y	N	N	N	N
311	N	N	Y	Y	N	N	N	N
412	Y	N	N	Y	Y	Y	N	N
512	Y	Y	Y	Y	Y	Y	N	N
612	N	N	N	Y	N	N	N	N
712	Y	N	N	Y	Y	Y	N	N
825	Y	Y	Y	Y	Y	Y	N	N
926	Y	Y	Y	Y	Y	Y	N	N
1028	Y	Y	Y	Y	Y	Y	N	N
1131	N	N	Y	N	N	N	Y	Y
1232	N	Y	Y	N	N	N	Y	Y
1333	N	N	N	N	N	N	Y	Y

Fig. 12. A snapshot of the requirement linking matrix generated by the model.

Table 3
Degree of link summary generated by our tool compared with the SME's linking.

Details	SME Link Level = L Total Links = 25 Overlap Range = 40% < OL <= 70%	SME Link Level = H Total Links = 14 Overlap Range = 70% < OL <= 100%
True positive	20	14
False positive	5	0

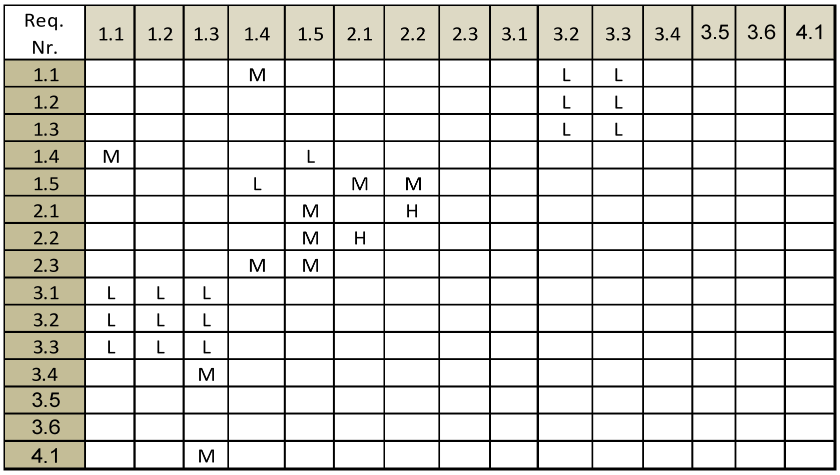
Table 4
Summary of the 2×2 requirement linking table.

False positives = the degree of links by the tool between the requirements varies from SME degree of linkage	5
False negatives = the tool rejected requirements as "No" link where SME linked them	0
True positives = the tool linked the requirements with the same level as the SME	34
True negatives = the tool rejected requirements as "No" link same as SME	0

the two requirements as having no-link, whereas the tool marked them as the linked. Fig. 13 reflects the partial mental model of the SME from the first approach.

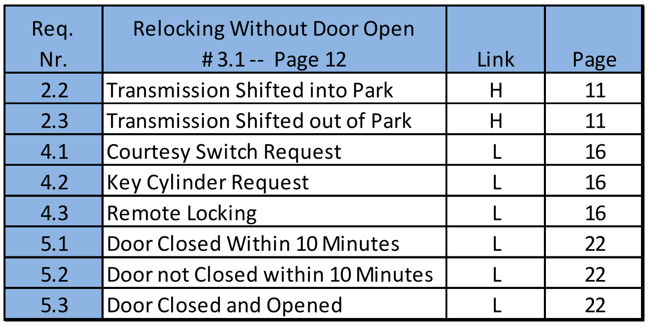
In the second approach the SMEs were asked to identify a small set of requirements and only those requirements were compared with all other requirements in a sequential manner. This comparison was done iteratively for all the sets formed by the SMEs. More effective results were achieved in this approach because only one set of requirements were compared with the others and therefore the focus was much better with minimized difficulty and the tediousness in terms of the back and forth traversing of the document. Figs. 14a and 14b shows linking performed by the SMEs for two such set of requirements compared with other requirements from the different pages. For example, in Fig. 14a the requirement #3.1 was highly linked with the requirements #2.2 and #2.3, whereas the same requirement scored low linking with the requirements #4.1, #4.2, and #4.3. Note that these requirements were spread over several pages of the documents. This figure indicates much clear focus in the review process by means of consistent linking information over several pages.

However, even this approach eventually leads to fatigue due to its iterative nature and because the requirements spanned across large number of pages (about 100 pages). The newly proposed model successfully handled all the requirements in one pass without missing any information. For instance, in one of such auto



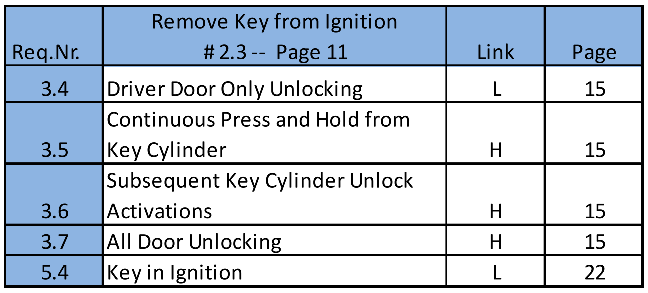
Req. Nr.	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	3.1	3.2	3.3	3.4	3.5	3.6	4.1
1.1				M						L	L				
1.2										L	L				
1.3										L	L				
1.4	M				L										
1.5				L		M	M								
2.1					M		H								
2.2					M	H									
2.3				M	M										
3.1	L	L	L												
3.2	L	L	L												
3.3	L	L	L												
3.4			M												
3.5															
3.6															
4.1			M												

Fig. 13. The partial mental model of the subject matter expert of Approach 1.



Req. Nr.	Relocking Without Door Open # 3.1 -- Page 12	Link	Page
2.2	Transmission Shifted into Park	H	11
2.3	Transmission Shifted out of Park	H	11
4.1	Courtesy Switch Request	L	16
4.2	Key Cylinder Request	L	16
4.3	Remote Locking	L	16
5.1	Door Closed Within 10 Minutes	L	22
5.2	Door not Closed within 10 Minutes	L	22
5.3	Door Closed and Opened	L	22

Fig. 14a. Sample of SME's mental model of Approach 2.



Req.Nr.	Remove Key from Ignition # 2.3 -- Page 11	Link	Page
3.4	Driver Door Only Unlocking	L	15
3.5	Continuous Press and Hold from Key Cylinder	H	15
3.6	Subsequent Key Cylinder Unlock Activations	H	15
3.7	All Door Unlocking	H	15
5.4	Key in Ignition	L	22

Fig. 14b. Sample of SME's mental model of Approach 2.

linking instances, two different requirements mentioned on page 16 (it was talking about "Door Locking") and on page 26 (it was talking about "Door Un-Locking"), which were marked by the tool as having a "Low" link, while the SMEs marked them as having "No-Link". When such requirements were presented to the SMEs a critical link was identified between these two requirements.

7. Conclusion and future work

In this paper, we have discussed a novel approach of linking the unstructured textual requirements for improved requirements and effective software development process using the semantic similarity model. Our model makes use of the multi-phrase terms identified from different requirement documents and these terms are used to compute the similarity score between the different requirements. Accordingly, the requirements are classified as having either "High", "Low" or No Link" between them. Our methodology relieves the reviewer from creating and updating a mental model by manually reading the lengthy requirement documents. This in turn makes sure that complete coverage is achieved without having to worry about the mental fatigue involved in the manual review process. The side benefit, but also a key advantage of the tool is that it provides time independency. In other words, the reviewer can split the review process over time as he/she is not required to keep a mental model alive while reviewing, because the tool provides the mapping. Our tool is deployed as a prototype and its performance has confirmed its usefulness when used by different SMEs.

In the future, our aim is to extend this framework in which the multi-term phrases will be constructed automatically from the requirement documents that are written not only in the English language, but also in Spanish, Korean, and Chinese. Moreover, we also aim at deriving the ontologies automatically from the software code such that the consistency checks between different requirements documents and that of between the requirements and their corresponding software can be performed automatically.

CRediT authorship contribution statement

Dnyanesh Rajpathak: Conceptualization, Methodology, Software, Investigation, Validation, Writing – original draft, Writing – review & editing. **Prakash M. Peranandam:** Conceptualization, Methodology, Software, Investigation, Validation, Writing – original draft, Writing – review & editing. **S. Ramesh:** Writing – original draft, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The Authors would like to thank Dr. John Cafeo, Dr. Jonathan Owen, and GM's publication review committee for their valuable suggestions and feedback, which helped to improve the quality of this paper.

References

Abadi, A., Nisenson, M., Simionovici, Y., 2008. A traceability technique for specifications, In: In Proc. of the 16th Inter. Conf. on Prog. Compre, Vol. 10, pp. 3–112.

Apostu, S., Burkacky, O., Deichmann, J., Doll, G., 2005. Automotive Software and Electrical/Electronic Architecture: Implicaions for OEM. McKinsey & Company.

Banerjee, S., Pedersen, T., 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In: Proc. of the Third International Conference on Intelligent Text Processing and Computational Linguistics, Mxico City.

Benedittini, O., Baines, T.S., Lightfoot, H.W., 2009. Greenough, R.M. State-of-the-art in integrated vehicle health management. J. Aerosp. Eng. 223 (2), 157–170.

Biemann, C., Bordag, S., Quasthoff, U., 2004. Automatic acquisition of paradigmatic relations using iterated co-occurrences. In: Proc. of 4th International Conferene on Language Resources and Evaluation, Lisboa, Portugal.

Blei, D., Ng, A., Jordan, M., 2003. Latent Dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.

Borg, M., Runeson, P., Ardo, A., 2014. Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. Int. J. Empir. Soft. Eng. 19 (6), 1565–1616.

Borg, A., Yong, A., Carlshamre, P., Sandahl, K., 2003. The bad conscience of requirements engineering: An investigation in real-world treatment of non-functional requirements. In: Proc. of the thrid conf. on Soft. Engg. Res. and Prac., Sweden, Lund, pp. 1-8.

Budanitsky, A., Hirst, G., 2006. Evaluating WordNet-based measures of semantic distance. Comput. Linguist. 32 (1), 13–47.

Capobianco, G., De Lucia, A., Oliveto, R., Panichella, A., Panichella, S., 2013. Improving IR-based traceability recovery via nounbased indexing of software artifacts. J. Softw. Maint. Evolut. Res. Pract. 25 (7), 743–762.

Cilibrasi, R., Vitanyi, P., 2007. The google similarity distance. IEEE Trans. Knowl. Data Eng. 19 (3), 370–383.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., 1990. Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. 41 (6), 391–407.

Dekhtyar, A., Huffman, H.J., Sundaram, S., Holbrook, A., Dekhtyar, O., 2007. Technique integration for requirements assessment. In: In Proc. of the 15th Inter. Requir. Engg. Conf, Vol. 14, pp. 1–152.

Devlin, J., Ming-Wei, C., Kenton, L., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistic, Minneapolis, Minnesota, Vol. 417, pp. 1–4186.

Egyed, A., 2001. Scalable consistency checking between diagrams - The VIEWINTEGRA Approach. In: Proc. of the 16th IEEE Inter. Conf. on Aut. Soft. Engg. IEEE Computer Society. p. 387.

Egyed, A., 2006. Instant consistency checking for the UML. In: Proc. of the ACM 28th Int. Conf. on Soft. Engg. Shanghai, China. Vol. 38, pp. 1-390.

Farfeleder, S., Moser, T., Krall, A., Stalhane, T., Omoronvia, I., Sojer, H., 2011. Ontology-driven guidance for requirement elicitation. In: Proc. of 8th Extd. Sema. Web Conf. (ESWC 2011). Crete, Greece, May 29-June 2 2011. Vol. 21. pp. 2-226.

Gabrilovich, E., Markovitch, S., 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proc. of the International Joint Conference on Artificial Intelligence, Vol. 160. pp. 6–1611.

Geravasi, V., Zowghi, D., 2005. Reasoning about inconsistencies in natural language requirements. ACM Trans. Softw. Eng. Methodol. 14, 277–330.

Gnesi, S., Lami, G., Trentanni, G., 2005. An automatic tool for the analysis of natural language requirements. Int. J. Comput. Syst. Sci. Eng. 20, 53–61.

Harispe, S., Sanchez, D., Ranwez, S., Janaqi, S., Montmain, J., 2014. A fremarowkr for unifying ontology-based semantic similarity measures: A study in the biomedical domain. J. Bio. Inf. 48, 38–53.

Huffman-Hayes, J., Dekhtyar, A., Osborne, J., 2003. Improving requirements tracing via information retrieval. In: Proc. of the International Conference on Requirements Engineering, Vol. 13. pp. 8–147.

Kamalrudin, M., Grundy, J., Hosking, J., 2010. Managing consistency between textual requirements, abstract interactions and essential use cases. In: Seoul, S. Korea (Ed.), Proc. of Intl. Conf. on Comp. Soft. App. (COMPSAC 2010). Vol. 32. pp. 7–336.

Kotonya, G., Sommerville, I., 1998. Requirements Engineering: Processes and Techniques. John Wiley Sons, Inc, ISBN: 0471972088.

Kozlenkov, A., Zisman, A., 2002. Are their design specifications consistent with our requirements? In: Proc. of the IEEE Joint Int. Conf. on Reqs. Engg. pp. 145-154.

Leacock, C., Chodorow, M., 1998. Combining local context and WordNet similarity for word sense identification. In: Fellbaum, C. (Ed.), WordNet: An Electronic Lexical Database. MIT Press, pp. 265–283.

Li, Y., Bandar, Z.A., McLean, D., 2003. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans. Knowl. Data Eng. 15 (4), 871–882.

Lormans, M., Deursen, A.V., 2005. Reconstructing requirements coverage views from design and test using traceability recovery via LSI. In: Proc. of the 3rd Inter. Workshop on Trace. In Emerg. Forms of Soft. Engg. Long Beach, California.

Mahmoud, A., Niu, N., 2015. On the role of semantics in automated requirements tracing. Requir. Eng. 20, 281–300.

Manning, C., Raghavan, P., Schutze, H., 2008. Introduction to Information Retrieval. Cambridge University Press.

Marcus, A., Maletic, J., 2003. Recovering documentation-to-source-code traceability links using latent semantic indexing. In: Proc. of the 25th Inter. Conf. on Soft. Engg. Vol. 12. pp. 5–135.

Mich, L., Franch, M., Inverardi, P.N., 2004. Market research on requirements analysis using linguistic tools. Requir. Eng. 9 (1), 40–56.

Mihalcea, R., Corley, C., Strapparava, C., 2006. Corpus-based and knowledge-based measures of text semantic similarity. In: In Proc. of the 21st Nat. Conf. on Artif. Intell., Vol. 1. AAAI Press, pp. 775–780.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word represen-tations in vector space. In: Proc. of the International Conference on Learning Representations.

Mohammad, S., Hirst, G., 2012. Distributional measures of semantic distance: a survey. arXiv preprint arXiv:1203.1858, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.337.9413&rep=rep1&type=pdf.

Nentwich, C., Emmerich, W., Finkelstein, A., Ellmer, E., 2003. Flexible consistency checking. ACM Trans. Softw. Eng. Methodol. 12, 28–63.

Noack, T., 2013. Automatic linking of test cases and requirements. In: Proc. of the Fifth International Conference Advances in System Testing and Vlidation Lifecycle, Venice, Italy.

Palmer, J.D., 1997. Software Requirements Engineering. IEEE Computer Society Press, New York,

Parvathy, A.G., Vasudevan, B.G., Balakrishnan, R., 2008. A comparative study of document correlation techniques for traceability analysis. In: Proc. of the 10th Inter. Conf. on Enter. Infor. Syst. Infor. Syst. Ana. and Spec, Vol. 6. pp. 4–69.

Patwardhan, S., Banerjee, S., Pedersen, T., 2003. Using measures of semantic relatedness for word sense disambiguation. In: Proc. of the 4th International Conference on Computational Linguistics and Intelligent Text Series, Mexico City, Mexico, Vol. 24. pp. 1-257.

Pennington, J., Socher, R., Manning, C., 2014. GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP. Vol. 153. pp. 2-1543.

Pohl, K., 1997. Change management. In: Mertens, P., Back, A., Becker, J. (Eds.), Lexikon Der Wirtschaftsinformatik. Springer, Berlin, Heidelberg.

Port, D., Nikora, A., Hayes, J.H., Huang, L., 2011. Text mining support for software requirements: traceability assurance. In Proc. of the 44th Hawaii Int. Conf. on Syst. Scie. Hawaii.

Rada, R., Mili, H., Bicknell, E., Blettner, M., 1989. Development and application of a metric on semantic net. IEEE Trans.Syst. Man Cybern. 19 (1), 17–30.

Rajpathak, D., 2013. An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain. Comput. Ind. 64, 565–580.

Rajpathak, D., Singh, S., 2014. An ontology-based text mining method to develop D-matrix from unstrucutred text. IEEE Tras. Syst. Man Cybern. 44 (7), 966–977.

Rajpathak, D., Xu, Y., Gibbs, I., 2020. An integrated framework for automatic ontology learning from unstructured repair text data for effective fault detection and isolation in automotive domain. Comput. Ind. 123.

Ramesh, B., Stubbs, C., Powers, T., Edwards, M., 1997. Requirements traceability: Theory and practice. Ann. Soft. Eng. 3, 397–415.

Ratnaparkhi, A., 1996. A maximum entropy part-of-speech tagger. In: Brill, E., Church, K. (Eds.), Proc. of the Conference on Empirical Methods in Natural Language Processing, Vol. 13, Philadelphia, PA. pp. 3–142.

Riloff, E., Shepherd, J., 1997. A corpus-based approach for building semantic lexicons. In: Proc. of the Second Conference on Empirical Methods in Natural Language Processing. Brown University, Providence, Rhode Island, pp. 117–124.

Roark, B., Charniak, E., 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In: Proc. of the 36th Annual Meeting of the Association for Computational Linguistics. Universit de Montreal, Montreal, Quebec, Canada, pp. 1110–1116.

Rosario, B., 2000. Latent semantic indexing: an overview. In: INFOSYS 240 Spring Paper. University of California, Berkeley.

Spärck, Jones K., 1972. A statistical interpretation of term specificity and its application in retrieval. J. Doc. 28, 11–21.

Sultanov, H., Hayes, J.H., Kong, W.-K., 2011. Application of swarm techniques to requirements tracing. Requir. Eng. 16 3, 209–226.

Toutanova, K., Manning, C.D., 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000). Vol. 6. pp. 3-70.

Turney, P.D., 2001. Mining Web for synonyms: PMI-IR versus LSA on TOEFL. In: Proc. of the 12th Eur. Conf. on Mach. Lear. ECML-2001, Freiburg, Germany, Vol. 49. pp. 1-502.

Turney, P.D., 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In: Proc. of the 40th Ann. Meeting of the Assoc. for Comp. Linguistics, Vol. 41. pp. 7–424.

Turney, P.D., 2006. Similarity of semantic relations. Comput. Linguist. 32 (3), 379–416.

Wang, W., Niu, N., Liu, H., Niu, Z., 2018. Enhancing automated requirements traceability by resolving polysemy. In: Proc. of the IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, Canada. pp. 40–51. http://dx.doi.org/10.1109/RE.2018.00-53.

Wong, W.E., Debroy, V., Restrepo, A., 2010. The role of software in recent catastrophic accidents. IEEE Trans. Reliab. 59 (3), 469–473.

Wu, Z., Palmer, M., 1994. Verb semantics and lexical selection. In: Proc. of the 32nd Ann. Meeting of the Assoc. for Comp. Linguistics, Morristown, NJ, USA, Vol. 13. pp. 3–138.

Xu, Y., Rajpathak, D., Gibbs, I., Klabjan, D., 2020. Automated ontology learning from domain-specific short unstructured text data. In: Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Vol. 3: KMIS. Vol. 2. pp. 9–39. ISBN: 978-989-758-474-9.

Zesch, T., Gurevych, I., 2007. Analysis of the wikipedia category graph for NLP applications. In: Proc. Of the TextGraphs-2 Workshop (NAACL-HLT 2007), Rochester, NY. pp. 1-8.

Zowghi, D., Gervasi, V., 2003. On the interplay between consistency, completeness, and correctness in requirements evolution. Inf. Softw. Technol. 45, 993–1009.

Dr. Dnyanesh G. Rajpathak is a Staff Researcher in General Motors. He obtained his Ph.D. degree in Artificial Intelligence from the Open University, UK, in 2004. He has over 15 patents and over 37 trade method secrets granted along with over 26 technical publications to his credit. He has been awarded with 2010 Charles L. McCuen award (GM R&D), GM's most prestigious 2011 "Boss Kettering" award, and General Motors President Award, for the work in data mining, anomaly detection, and text mining. Currently, he is an Associate Editor of SAE International Journal of Aerospace. He has served as a Program Committee member of different conferences ACM CIKM 2020-21, ESWC'20, ESWC'14, ESWC'13, PDCTA-2016, and served as a reviewer in several IEEE and other international journals as well as conferences. His current research interests include deep learning, machine learning, data and text mining for diagnosis and prognosis, root cause investigation, emerging safety issue detection, integrated vehicle health management and manufacturing 4.0.

Dr. Prakash M. Peranandam is a senior researcher in GM R&D. He obtained his Ph.D. in 2006 from the University of Tuebingen, Germany. He has co-authored more than of 30 research publications, technical reports and generated over 40 IPs. His research interests are in requirements analysis & engineering, virtualization, verification & validation (V&V) of Embedded software and AV/ADAS systems.

Dr. Ramesh S has been with General Motors Global R&D where he currently holds the position of Senior Technical Fellow and thrust area lead for model based embedded software. At General Motors, he is responsible for providing technical leadership for research and development in several areas related to Electronics, Control & Software processes, methods, and tools. His broad areas of interests are Rigorous Software Engineering, Embedded Systems and Real-Time Systems. He is the author of several patents and has published more than 100 papers in peer-reviewed International journals and conferences. He is on the editorial boards of the International Journal on Real-Time Systems, Eurasip Journal on Embedded Systems, and earlier on IEEE Journal on Embedded System Letters. Prior to joining GM R&D, he was on the faculty of the department of Computer Science & Engineering at IIT Bombay, for more than fifteen years. At IIT Bombay, he played a major role in setting up a National Centre for Formal Design and Verification of Software. As the founding head of this Centre, he carried out many projects on verification of embedded software, for several Government organizations. He is a fellow of the Indian National Academy of Engineering and was visiting/adjunct faculty of many institutions.