IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 October 2023, accepted 3 January 2024, date of publication 16 January 2024, date of current version 1 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354955

RESEARCH ARTICLE

Detecting Pragmatic Ambiguity in Requirement Specification Using Novel Concept Maximum Matching Approach Based on Graph Network

KHADIJA ASLAM[1], FAIZA IQBAL[2], AYESHA ALTAF[2], NAVEED HUSSAIN[3], MÓNICA GRACIA VILLAR[4,5,6], EMMANUEL SORIANO FLORES[4,7,8], ISABEL DE LA TORRE DÍEZ[9], AND IMRAN ASHRAF[10]

[1]Strategic Systems International, Lahore 54000, Pakistan
[2]Department of Computer Science, University of Engineering and Technology (UET), Lahore 54890, Pakistan
[3]Department of Software Engineering, University of Central Punjab, Lahore 54700, Pakistan
[4]Universidad Europea del Atlántico, 39011 Santander, Spain
[5]Universidad Internacional Iberoamericana Arecibo, Sector Palaches 00613, Puerto Rico
[6]Universidade Internacional do Cuanza, Kuito, Bié, Angola
[7]Universidad Internacional Iberoamericana, Campeche 24560, Mexico
[8]Fundación Universitaria Internacional de Colombia, Bogotá 111321, Colombia
[9]Department of Signal Theory, Communications and Telematics Engineering, Unviersity of Valladolid, 47011 Valladolid, Spain
[10]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea

Corresponding authors: Faiza Iqbal (faiza.iqbal@uet.edu.pk) and Imran Ashraf (ashrafimran@live.com)

This research was supported by the European University of Atlantic.

ABSTRACT Requirements specifications written in natural language enable us to understand a program's intended functionality, which we can then translate into operational software. At varying stages of requirement specification, multiple ambiguities emerge. Ambiguities may appear at several levels including the syntactic, semantic, domain, lexical, and pragmatic levels. The primary objective of this study is to identify requirements' pragmatic ambiguity. Pragmatic ambiguity occurs when the same set of circumstances can be interpreted in multiple ways. It requires consideration of the context statement of the requirements. Prior research has developed methods for obtaining concepts based on individual nodes, so there is room for improvement in the requirements interpretation procedure. This research aims to develop a more effective model for identifying pragmatic ambiguity in requirement definition. To better interpret requirements, we introduced the Concept Maximum Matching (CMM) technique, which extracts concepts based on edges. The CMM technique significantly improves precision because it permits a more accurate interpretation of requirements based on the relative weight of their edges. Obtaining an F-measure score of 0.754 as opposed to 0.563 in existing models, the evaluation results demonstrate that CMM is a substantial improvement over the previous method.

INDEX TERMS Pragmatic ambiguity, natural language, requirements specification, knowledge base, ambiguity detection.

I. INTRODUCTION

Requirement engineering (RE) process facilitates the creation of consumer-centric products. Utilizing RE activities captures the complexities of requirements. These activities include requirement analysis, requirement elicitation, requirement specification, requirement validation, and requirement management [3]. Typically, requirements are described using natural language, which can result in ambiguity. Ambiguity is the misinterpretation of requirements that results in

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

TABLE 1. Research gap in existing studies.

Existing Techniques	Research Gap
Using an extended path to achieve more relatedness of requirements interpretations. [1]	Lack of providing meaningful interpretations of requirements
Using the shortest path algorithm based on the single term (node) to requirements interpretation. [2]	Only provides a single node-based shortest path for the requirement.

a product that does not meet the expectations of the stakeholders. Establishing clear and accurate connections between requirements can lead to the development of a product that is both accurate and meets the necessary criteria for acceptance Numerous approaches help bridge the gap between informal and formal requirements specification, such as Object Constraint Language (OCL), B method, Z language, and Vienna Development Method (VDM), among others [4], [5]. However, further enhancements are necessary to detect requirement ambiguities. Detecting ambiguities at an early stage is the greatest challenge that can help reduce time, effort, and cost. Typically, requirements are specified in natural language, resulting in several linguistic ambiguities [6]. These ambiguities are classified as syntactic, semantic, lexical, and pragmatic ambiguity, respectively [7].

The sentence structure determines syntactic ambiguity. It is also referred to as structural ambiguity and occurs when a given sentence can be interpreted in multiple ways [8]. Consider the sentence "large toy factory" as an example. This case provides two interpretations, including (big toy) factory and big (toy factory) [9]. Lexical ambiguity arises when a word possesses multiple meanings or when different origins employ the identically spelled word in written and spoken language. For instance, the word "bank" can mean both *financial institution* and *riverbank*. Semantic ambiguity arises when a sentence's meaning can be constructed in numerous ways depending on its context [9]. Consider the statement, "Every administrative member has a system login number". There are two ways to interpret the context of this sentence: either *every administrative member has a unique system login number*, or, *every administrative member has the same system login number*. In pragmatic ambiguity, a sentence can have multiple context-dependent meanings. True meaning interpretation requires domain, situation, background knowledge, and contextual understanding [10]. Consider the phrase "User access will be restricted/blocked due to violation." Here, pragmatic ambiguity arises in the form of the question, *What mechanism will be used to block user access?* IP/MAC address/user account bans, for example? It has been determined that existing approaches do not account for all concepts and therefore cannot achieve greater precision. Similar kinds of ambiguities can be detected while analyzing product recommender systems [11] and spam reviews [12]. Comparing existing studies and highlighting research gaps, Table 1 illustrates the research problem.

Practical ambiguity detection enhances comprehension of the actual requirement. In requirement specifications, such pragmatic ambiguity may arise, leading to the presentation of a single requirement's concept in multiple ways. The purpose of this research is to identify pragmatic ambiguity in requirement specifications using a knowledge base model as a foundation [1], [2].

Existing approaches utilize the shortest path technique, which activates highly weighted concepts in a weighted graph to achieve requirements interpretation. Activating heavily weighted concepts in graphs indicates that concepts in domain documents are more related. Based on a couple of neighboring nodes with higher occurrences in domain knowledge graphs, the existing shortest path algorithm retrieved more matching results. Significant inaccuracy in requirement interpretation is the limitation of these techniques. It requires additional enhancements to detect pragmatic ambiguity to improve the accuracy of the requirements interpretation process and generate meaningful interpretations. It is necessary to design an approach that can retrieve concepts based on multiple nodes and their edges to effectively detect pragmatic ambiguity and improve overall precision.

The proposed model extended the existing graph-based modeling approach that is based on the shortest path algorithm. Due to its reliance on single-node matching, the existing method lacks a high degree of precision. This study proposed a Concept Maximum Matching (CMM) method based on edges and nodes. CMM retrieves concepts based on edges to achieve a more precise interpretation of requirements. A set of requirements and multiple domain documents are inputs for the designed method. The requirements are exhaustively researched in the available domain documents, resulting in a variety of interpretations from each domain document. The output represents the similarity between the retrieved interpretations, which is used to determine whether a given set of requirements is ambiguous. The greater similarity between retrieved interpretations indicates requirements with no ambiguity, and vice versa. The proposed CMM method overcomes the shortcomings of the existing method and achieves a higher accuracy score. Regarding the pragmatic interpretation of requirements, the operation of the proposed CMM approach has been compared to existing approaches. This study makes the following contributions

1) Proposed CMM approach which utilized multiple nodes and edges of concept knowledge graphs,
2) Developed an algorithm to build concept-based knowledge graphs using domain documents,
3) Evaluated and analyzed the effectiveness of the CMM approach in efficiently detecting pragmatic ambiguity in requirement specifications and improving the accuracy of the requirement interpretation process

The rest of the paper is organized as follows. The introduction section is followed by the literature review Section II which describes existing work on ambiguity

TABLE 2. Summary of existing literature in the domain of ambiguity detection.

Ref.	Ambiguity	Techniques	Accuracy	Limitation
[1]	Pragmatic	Graph-based modelling knowledge	Precision 51% and recall 63%	Built strong influence but lacks in refinement
[13]	Pragmatic	Knowledge Dictionary	Recall 99%	Neglect words weight based on their frequency of occurrences.
[14]	Pragmatic	QoI-awareness	-	Only gives better result against single query only.
[15]	Grammar	Artificial Neural Network	-	Lack in quality
[16]	Domain	Domain-Specific Corpora	Precision 80% and recall 89%	Only works with Syntactic ambiguity and inefficient for large data set
[17]	–	NLP, Word embedding	Precision 83% and recall 85% -	Limited data size
[18]	Grammar	Word Segmentation Based Dictionary	-	Limited accuracy
[19]	Semantic	Semantic annotation	Precision 96.6% and recall 96%	The approach utilized controlled natural language to collect requirements and did not address way to detect ambiguity in natural language requirements.
[21]	Semantic	Conflict detection	-	Failed to consider conflict based on duplicate requirement
[22]	NL ambiguities	TaskLint	-	Insufficient to detect contextual information.
[23]	Anaphoric	BERT	Precision 60% and recall 100%	Just able to examine Anaphoric ambiguity
[24]	Lexical	Controlled Language	Recall 80.12 % and Precision 85.76 % S	Still unable to remove totally ambiguities.
[25]	Syntactic	Filtering pipelines,Stanford	Precision 65 % and recall 99 %	Inefficient for non-ambiguous requirements.
[26]	Grammar	Unsupervised learning	-	Lack to target total quality of MT

detection in requirement documents. Section III presents the proposed CMM approach and discusses its various important elements. This section demonstrates domain knowledge graph representation and the weighted graph construction process as well. The evaluation of the proposed approach is presented in Section IV. This section also describes a comparative analysis of CMM with existing approaches. Finally, Section V concludes the paper and presents future directions.

II. LITERATURE REVIEW

There are a variety of classification approaches for ambiguities in natural language [26]. Lexical ambiguity, semantic ambiguity, pragmatic ambiguity, and syntactic ambiguity are the primary types of ambiguity. In lexical ambiguity, a single word has multiple meanings. It occurs when two different origins use the same written and spoken spellings of a word [27]. Structure ambiguity is another name for syntactic ambiguity, which depends on sentence structure-based ambiguity. It occurs when a given sentence can be interpreted in multiple different ways [28], [29]. There is semantic ambiguity when the meaning of a sentence can be interpreted in multiple ways based on its context [30]. While in pragmatic ambiguity, a sentence can have multiple contextual meanings [31], [32].

It requires domain, situation, context, and background knowledge to comprehend the actual meaning. Detection and interpretation of pragmatic ambiguity in requirement specification documents can maximize comprehension of the actual requirement. The impact of pragmatic ambiguity in requirement specification documents has not been exhaustively analyzed in the existing literature. This kind of ambiguities can be highlighted in diverse domains e.g. anomaly detection and secure networks [33], [34], [35], [36], blockchain models [35], [37], [38]. Table 2 discusses pragmatic ambiguity detection approaches in requirement documents. Existing studies on the concept of pragmatic ambiguity detection, employed techniques, and existing limitations are elaborated and analyzed.

Different computerized tools of pragmatics are used in natural language tasks [39], so the detection of pragmatic ambiguity is crucial to get precise responses. The research [40] utilizes word embedding algorithms to discover and recognize ambiguous terms within requirements. Additionally, connected data is used to address the identified ambiguities. The efficiency of the suggested tool was evaluated using open-source software specification papers. The performance of the tool in detecting and correcting ambiguity was compared to that of people. Similarly, the authors [21] propose TaskLint, a system that identifies errors with task instructions automatically. TaskLint employs a variety of existing natural language processing (NLP) methods to recognize words and phrases that may indicate worker uncertainty. This method is comparable to code analysis tools known as "linters" that identify code characteristics that may signal the presence of flaws. The evaluation of TaskLint utilizing novice-created task instructions validates the ability of static analysis techniques to increase task clarity and boost result correctness. Nonetheless, the review identifies a number of obstacles that must be addressed.

Based on an analysis of existing literature and prevalent limitations, it has been determined that five different approaches are used to detect pragmatic ambiguity such as shortest-path search algorithm [1], [2], machine learning

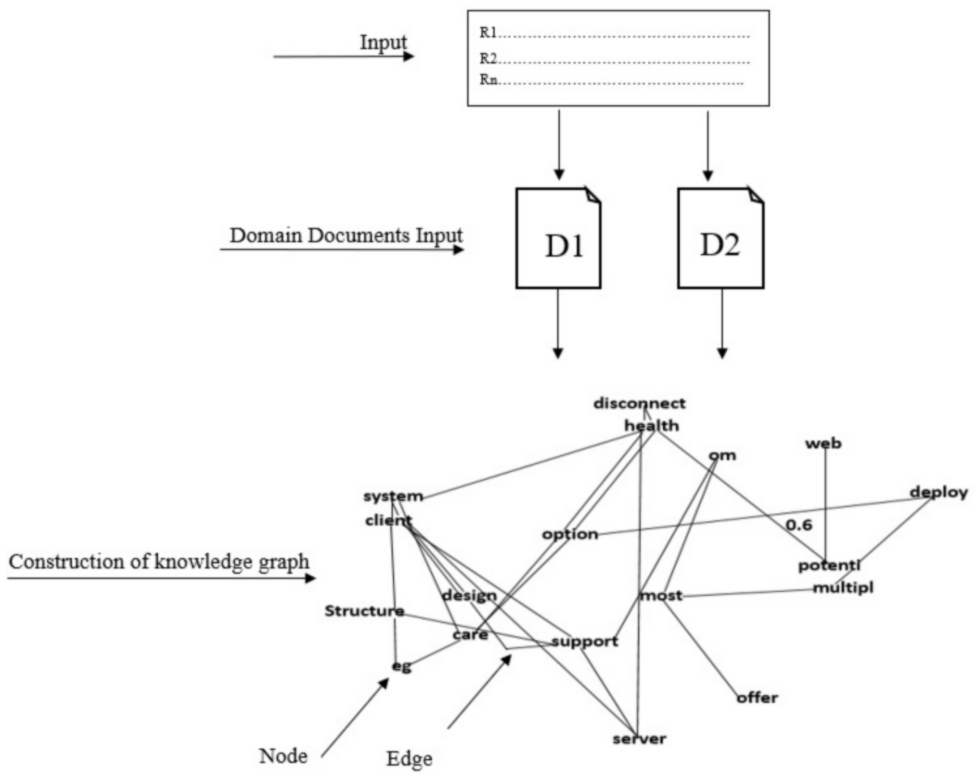


FIGURE 1. Working flow of the proposed CMM approach.

and heuristics [8], [40], word embedding [41], knowledge base approach [13] and controlled language [8], [40]. The precision and recall of most of these existing techniques are provided in Table 2. It is necessary to develop a method for detecting pragmatic ambiguity to remove the complexity of requirements based on connecting related concepts using multiple concepts matching [42]. It is necessary to have a domain knowledge graph with weighted edges to utilize available connecting paths.

III. PROPOSED CONCEPT MAXIMUM MATCHING APPROACH

The CMM provides the greatest similarity and relationship between concepts. We have modeled the CMM algorithm using edge weight. This method retrieves concepts based on their edges for improved interpretation of requirements. As stated previously, pragmatic ambiguity focuses on the context specification of the requirements; therefore, the interpretation of the requirements is contingent on the reader's prior knowledge. Based on their prior knowledge, various readers interpret a concept differently. Existing methods [1], [2] generate artificial graph-based subjects to model background and domain knowledge. After constructing the domain knowledge graph (using the provided domain documents), the shortest path is used to determine the true relationship between the given requirements. To achieve the interpretation of requirements, the shortest path activates heavily weighted concepts in a weighted graph. Activating heavily weighted concepts in graphs indicates that concepts in domain documents are more related. Existing studies were designed to retrieve concepts based on single nodes, which failed to achieve improved accuracy, necessitating additional enhancements to the requirements interpretation process.

In comparison to the existing shortest path algorithm, which retrieved matching results based on a couple of neighboring nodes with higher occurrences in domain knowledge graphs, the proposed algorithm retrieves matching results based on a larger number of nodes with higher occurrences. The proposed CMM method retrieves concepts based on edges for improved interpretation of requirements. Figure 1 depicts the workflow of the proposed CMM method for determining pragmatic ambiguity. CMM accepts a set of requirements and multiple domain documents as input, with the requirements being looked up in the domain documents. It then determines whether the given set of requirements is ambiguous or unambiguous based on the degree of similarity between the retrieved interpretations. The greater similarity between retrieved interpretations indicates unambiguous requirements, whereas lower similarity values indicate ambiguous requirements [41].

A. DOMAIN KNOWLEDGE GRAPH REPRESENTATION

To construct a domain knowledge graph, Algorithm 1 requires domain documents as input. The designed algorithm returns a relational graph that includes all document concepts.

TABLE 3. Description of notations used for the model.

Notation	Description
G	Graph
V	Node
E	Edge(Vi,VJ) OR (node1,node2)
W	Weight of concept
T	Set of Terms or Node
Ω	Stop words
R	Requirement
R'	Set of stems
DD	Domain document
S	Sentence
X	Domain document

In a domain knowledge graph, nodes represent concepts while edges indicate the connection between nodes.

B. WEIGHTED GRAPH CONSTRUCTION

A weighted graph consists of nodes, edges, and weight of nodes $G = (V, E, W)$. These notations are described in Table 3.

1) NODES

All the terms used in domain documents are called nodes and are also known as concepts. Stop words like conjunctions, pronouns, articles, and prepositions are not included in a set of nodes. Stop words provide the structure of a sentence whereas we need to focus on the content of concepts due to which we have not considered the stop words. Node set is represented as $T = t1, t2, t3 \ldots\ldots tn$. $T = T - \Omega$, where Ω represents set of stop words. Moreover, we have transformed each term into a morphological root to retrieve basic concepts, for example, detection and detector. Here both of these terms share the same concept that is detected. The nodes function is labeled as $\$: V \rightarrow T$.

2) EDGES

Edges are used to connect different nodes and provide connections between them. These have co-occurrence in related nodes with each other inside the graph. Edge enables the direction of orders within connecting nodes. Edges in our graph are represented as $E = (vi, vj)$.

To understand this, consider an example: Here R represents a requirement and R' represents its concepts.

R= "Systems supporting OM should track the changes made to the status of case records as a result of changes in the case definition".

R' = {'system', 'support, 'OM, 'should, 'track, 'change, 'made, 'status, 'case, 'record, 'result, 'change', 'case',' definition'};

The co-occurrences of stem associated with edges in this graph are 'system', 'support'), ('support', 'OM'), ('OM', 'should'), ('track', 'change), ('change', 'made'), ('made', 'status'), ('status', 'case'), ('case', 'record'), ('record', 'result'), ('result', 'change'), ('change', 'case'), ('case', 'definition')}. In the directed graph direction of the edge set matters so ('system', 'support') and ('support', 'system') will be considered different.

Algorithm 1 : *BuildEdge − Graph*(D, Ω)

```
    Input : $(D, \Omega)$
    Output : $G$
1: $\overline{D}$,V,E,W $\leftarrow \emptyset$
2: $\overline{D} \leftarrow$ PRE-PROCESS (D,Ω)
3: $V \leftarrow$ BUILD-Edge-NODES $(\overline{D})$
4: $E, W \leftarrow$ BUILD-EDGES $(\overline{D})$
5: Return $G \leftarrow$ {V,E,W}
```

Algorithm 2 Pre-Process Algorithm (DD, Ω)

```
    Input : $(DD, \Omega)$
    Output : $\overline{DD}$)
1: $(\overline{DD}) \leftarrow \emptyset$
2: for S ϵ DD do
3:     $\overline{S} \leftarrow (s \notin \Omega)$
4:     $\overline{DD} \leftarrow \overline{DD} \cup \overline{S}$
5: end for
6: Return $\overline{DD}$
```

C. WEIGHT FUNCTION

The weight function is used to calculate the count of co-occurrences of nodes connected together with each other inside domain documents. We have a function called $\chi : T \times T \rightarrow \mathbb{N}$ that counts how often two nodes occur together.

Each connection between nodes in a graph has a number based on a weight function called $w(e) = \frac{1}{\chi(L(v_i), L(v_j))}$.

For any connection (vi, vj), the weight is calculated as $w(e) = 1$ divided by the number of times nodes $L(v_i)$ $L(v_j)$ appear together, which is represented by χ.

Every connection in the graph represents nodes that co-occur in the original documents. If there's a connection, it means those nodes appeared together at least once.

$\chi(L(v_i), L(v_j)) > 0$, is always greater than zero because an edge in the graph only exists when the two-word nodes it connects have been found together in the original set of documents.

D. KNOWLEDGE GRAPH ALGORITHM

The CMM approach has been represented using a knowledge graph algorithm that utilizes Edge-Graph, pre-process, build-nodes, sentence path search, and build-edge algorithms. Algorithm 1 represents a graph construction algorithm that describes steps used to produce an edge-graph representation from a given input.

Algorithm 2 takes domain document DD including stop words Ω. We have omitted all stop words, as we need to know the context rather than the structure of the sentence. Once stop words are omitted, obtained set of words that are used inside the document, are transformed into morphological roots. For example, the domain sentence is "Application shows products based on collaborative filters". This step of the algorithm transforms all sentences in the domain document in the following way.

S = {'Application', 'show, 'product, 'collaborate', 'filter' };

Algorithm 3 Build-Nodes Algorithm $\overline{DD}$

```
    Input : $\overline{DD}$     Output : $V$
1:  V $\leftarrow \emptyset$
2:  for $\overline{S} \in \overline{DD}$ do
3:      $\overline{s} \in \overline{S}$
4:      if $\overline{s} \notin$ V then
5:          V $\leftarrow V \cup \overline{s}$
6:      end if
7:  end for
8:  Return V
```

Algorithm 4 Build-Edges Algorithm ($\overline{DD}$)

```
Input: $\overline{DD}$
Output: E, W
1:  $E, W, X \leftarrow \emptyset$
2:  $V \leftarrow \emptyset$
3:  for $\overline{S} \in \overline{DD}$ do
4:      for $\overline{s}_h, \overline{s}_k \in \overline{S},\ k = h + 1$ do
5:          if $(\overline{s}_h, \overline{s}_k) \notin E$ then
6:              $e \leftarrow (\overline{s}_h, \overline{s}_k)$
7:              $E \leftarrow E \cup \{e\}$
8:              $W[e], X[e] \leftarrow 1$
9:          else
10:             $X[e] \leftarrow X[e] + 1$
11:         end if
12:     end for
13: end for
14: for $W[e] \in W$ do
15:     $W[e] \leftarrow \frac{1}{X[e]}$
16: end for
17: return E, W
```

After performing the first step, we have added each unique node in the set of vertices as used inside the document as represented in Algorithm 3. New stems found in the document will be updated in a set of vertices. This step of the algorithm scans all sentences to find unique nodes.

Algorithm 4 represents the last step of the construction domain knowledge graph which creates edges E to the graph and a weight vector W. For every pair of nodes in each sentence of the document set DD, an edge is formed in the graph. To calculate the weights in vector W, we use a support vector X. Each element in X corresponds to an edge and keeps track of how many times the connected nodes co-occur. When a new edge e is introduced, $X[e]$ is set to 1. If a co-occurrence is detected for which an edge already exists, the corresponding $X[e]$ is incremented by 1. In the final step of the process, we create the weight vector W by taking the reciprocal of each element in X.

Algorithm 4 takes domain documents $D1, D2, D3 \ldots .DN$ as an input. The domain document set is retrieved from different web search engines (e.g., Yahoo, Google Scholar, Bing, and Google). The relevancy of the domain documents is manually verified. Here we briefly describe the process of the algorithm to build a graph from domain documents automatically, all details are given in paper [2]. After taking documents, the stop words are omitted and each term is transformed into a morphological root to retrieve basic concepts (called stems). After that, obtain the set of stem construct graphs for all concepts, which are used in domain documents, and show the association among concepts with edges. Figure 2 shows the process of edges-based graph construction.

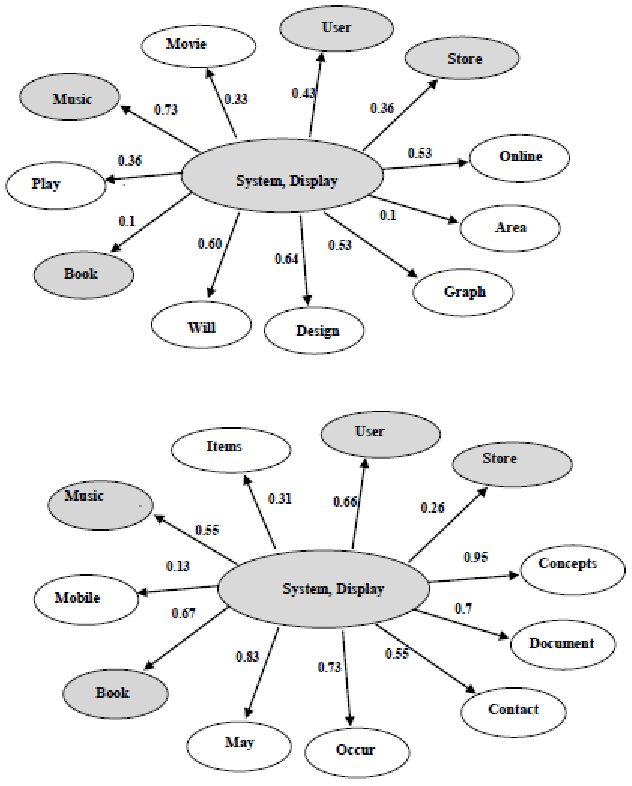


FIGURE 2. Building domain knowledge graph.

Figure 2 represents an example in which nodes and the association of edges are presented. Edges show the number of co-occurrences of terms with edges.

E. BUILD SHORTEST PATH BASED ON NODE AND EDGE

The complete working procedure of the CMM approach is given in Algorithm 5. It works in the following steps.

Step 1: UPDATE-NODES updates the newly found nodes of Requirement in existing graphs.

Step 2: BUILD-MIN-PATH returns all sub-paths identified by the shortest path algorithm as nodes that co-occur in related nodes within the graph. On finding a new edge, construct the path and compute the edge weights.

Step 3: BUILD-EXT-PATH extends with those paths that have a greater weight in the build-min-path algorithm. As co-occurrence increases, the relationship between nodes becomes stronger. Retrieves all newly created sub-paths that are direct with this path.

Algorithm 5 Sentence Path Search ($\overline{R}$, V, E, W, X)

```
**Input** : ( $\overline{R}$, V, E, W, X)
**Output**:EE
1: P ← ∅
2: EE ← $EdgesPath$
3: V ← $UPDATE-NODES(\overline{R}, V)$
4: P ← BUILD-MIN-PATH($\overline{R}$,V,E,W)
5: EP← BUILD-EXT-PATH (P, V, E, W)
6: EE ← BUILD-EDGES-PATH (P, V, E, W)
7: **Return** EE
```

UPDATE-NODES, BUILD-MIN-PATH and BUILD-EXT-PATH have been described in existing work [1], [2]. This work contributes to the previous work by Ferrari et al. [1], [2]. For BUILD-EDGES-PATH, Algorithm 6 represents the method for constructing the entire path of the interpretation process. This algorithm's final step accomplishes edge-based requirements interpretation. Here, we have interpreted the requirements using the shortest path based on the weight of the edges, which increases the accuracy of the interpretation process.

In Algorithm 6, tokens correspond to the document's nodes. When this token is combined with another token in a document, a connection in the form of an edge is created. Therefore, the significance of a pair of edges increases as it appears more frequently. We can determine whether a requirement is ambiguous or not based on the edge weights. To comprehend this procedure, consider a running example. Consider the following stipulation R.

R = "The system shall display similar books based on the user preferences of other users who purchased the same book during previous sessions".

R' = {'system', 'shall', 'display', 'similar', 'book', 'base', 'user', 'prefer', 'user', 'purchas', 'book', 'previou', 'session'};

Based on the approach, following combinations or paths are formed: P1(R) = {('system', 'shall'), ('display', 'similar'), 'user', 'may', 'avail', ('book', 'base'), 'movi', 'user', ('user', 'prefer'), ('user', 'purchas'), 'onlin', 'store', 'will', 'play', 'music', ('book', 'previou'),'movi', 'user', 'session'};

P2(R) = {('system', 'shall'),'drama, 'user' ('display', 'similar'), 'stories' ('book', 'base'), ('user', 'prefer'), 'item', 'will', 'play', 'music', ('user', 'purchas'), ('book' 'previou'), 'list', 'user', 'session'}.

Finally, we find the similarity between retrieved interpretations to decide whether the given set of requirements is ambiguous or unambiguous. {'user', 'will', 'play', 'music'} are common terms in both paths. More similarity between retrieved interpretations shows unambiguous requirements and vice versa. After using this technique [1], [2] to detect ambiguity Ferrari et al. [41] proposed another technique to detect pragmatic ambiguity for ranking ambiguous terms. This approach works to detect ambiguity in Cross-domain while our approach works for the same domain.

Algorithm 6 Build Edges Path (P, V, E, W)

```
**Input**: (P, V, E, W)
**Output**: update path weight
1: Tokens ← List of nodes
2: T ← Tokens
3: CT ← current Token
4: e ← CT, CT[i+1]
5: Net ← Nodes in complete documents
6: W ← Weight
7: CNewEdge ← CurrentNewEdge
8: **for** T **do**
9:     **if** Net.has_Edge(E) **then**
10:        X[e] = X[e] + 1
11:    **end if**
12: **end for**
13: (We have created a new list of edges) New
14: **for** CNewEdge **do**
15:    **if** CNewEdge % 2 == 0 **then**
16:        TrimmedEdgesList.add(NewEdge)
17:        **for** currentEdge in trimmedEdgesList **do**
18:            **if** currentEdge % 2 == 0 **then**
19:                Node2.add(currentEdge).add(Second node)
20:            **else**
21:                Node1.add(currentEdge).add(Second node)
22:            **end if**
23:        **end for**
24:        NewEdge = (Node1[i], Node2[i])
25:        (i.hasEdge in Net)
26:        (increase weight of edge i)
27:        Add update path weight
28:    **end if**
29: **end for**
```

IV. EXPERIMENTAL RESULTS

This section presents detailed information on the experimental evaluation of the proposed CMM approach. Table 4 represents the list of domain documents that are used in the experiments with different combinations of documents. The documents are taken from the same dataset that has been used in existing work [2]. The objective is to perform a fair comparison with existing works and differentiate between the performance of the proposed approach from existing approaches.

Table 5 represents the combinations of documents, which we have used in $k-th$ analysis for both existing and proposed approaches. We have evaluated the result three times with six documents with different combinations of documents to analyze the difference between the existing and proposed approaches.

A. EVALUATION OF RESULTS

First, we have performed $j-th$ analysis ($j = 1, \ldots, k$) with different combinations of documents to find the similarity value for each requirement ($R_i \epsilon R$). $j-th$ analysis provides values each time $\sigma^j_{Ri}(EE_1, EE_2, \ldots, EE_n)$. We combine

TABLE 4. Domain documents used in this study.

ID	Title	Link
D1	Document PHEMCE strategy	http://goo.gl/hYaipm
D2	Document Public Health Practice	http://goo.gl/hVVy1Y
D3	Document about control of outbreaks	http://goo.gl/Sv4Ebu
D4	Document epidemic preparedness	http://goo.gl/PK9yn7
D5	Document "Outbreak"	http://goo.gl/LUQEWm
D6	Document "Management System"	http://goo.gl/mgWfhh

TABLE 5. Combinations of documents for each k-th analysis.

K	G1
	G2
1	D1
	D2
2	D3
	D4
3	D5
	D6

$j-th$ analysis values by $\sum(R_i)$ to get an aggregate value. We compare the value of $\overline{\sigma}(R_i)$, $\sigma_{min}(R_i)$ with the threshold value to know if the requirement is ambiguous or not ambiguous. Values of $\sigma_{min}(R_i)$, $\overline{\sigma}(R_i)$ below the threshold indicate the presence of ambiguity. Two methods are selected to compute the value of $\sum(R_i)$: one for average value second for minimum value.

B. AVERAGE AND MINIMUM VALUE

The average value is calculated using Equation 1

$$\overline{\sigma}R_{\mathrm{i}} = \frac{\{\sigma^1(R_{\mathrm{i}}) + \sigma^2(R_{\mathrm{i}}) + \ldots + \sigma^k(R_{\mathrm{i}})\}}{k} = \sum(R_{\mathrm{i}}) \quad (1)$$

The minimum value is calculated using Equation 2

$$\sigma_{\min}(R_{\mathrm{i}}) = min\{\sigma_1(R_{\mathrm{i}}) + \sigma_2(R_{\mathrm{i}}) + \ldots + \sigma^k(R_{\mathrm{i}})\} = \sum(R_{\mathrm{i}}) \quad (2)$$

For all values of $\sum(\mathrm{R_i})$, where $(R_{\mathrm{i}} = I, \ldots, m)$ we have computed threshold value by arithmetic mean.

$$T = \frac{\{\sum(R_1) + \sum(R_2) + \ldots + \sum(R_{\mathrm{m}})\}}{m} \quad (3)$$

Here, $\overline{\sigma}$ calculated using Equation 1 and $\sigma_{\min}$ calculated using Equation 2 represent two different methods to analyze the ambiguity of requirements. To understand all of the evaluation methods in their proper context, a running example is provided below. A similar example has been considered in [1] and [2]. As an illustration, we have a list of five requirements to meet.

- R= "R1: Systems supporting OM should support multiple deployment options (e.g., client-server, disconnected, and potentially web-based)".
- R2: "Systems supporting OM should provide the ability for computers in disconnected mode to reconnect to a server to share OM data among other computers that operate in disconnected mode OM data should be synchronized so that all instances of OM applications working from the same server are able to share and use the same data".

TABLE 6. Combinations of documents for each k^{th} analysis.

R	G1
	G2
(R1,R2,R3,R4,R5)	D1
	D2
(R1,R2,R3,R4,R5)	D3
	D4
(R1,R2,R3,R4,R5)	D5
	D6

TABLE 7. k^{th} analysis values for each requirement.

K	R1	R2	R3	R4	R5	R6	R7
K1	0.324	0.214	0.104	0.114	0.365	0.224	0.104
K2	0.184	0.192	0.273	0.932	0.576	0.431	0.184
K3	0.280	0.104	0.430	0.510	0.840	0.432	0.104

- R3: "Systems supporting OM should be able to electronically record and store data from remote devices that may be uploaded to an aggregating system".
- R4: "Systems supporting OM should be capable of using configurable, domain-specific vocabulary".
- R5: "Systems supporting OM should have the ability to record the case definition for a health event".

We have performed k^{th} analysis three times ($J = 1, \ldots, k$) for each requirement R1, R2, R3, R4, R5 all details are mentioned in Table 6.

Table 6 depicts the requirements sets that we used for each iteration in conjunction with a variety of documents to determine whether the requirements are ambiguous. Each iteration utilized the same requirement sets, but the documents were combined in a variety of ways.

Table 7 shows the k^{th} analysis values for each requirement $R1$ to $R7$. The threshold for minimum and average value is 0.1306 while the average value is 1.0884. After computing the Threshold, we need to calculate precision and recall. Precision and recall metrics are employed to find the accuracy of results. Total requirements are 114 and manually predicted requirements marked as ambiguous are 43. We have computed threshold ($\Gamma\sigma$min 0.33552, average 0.51006) by three analyses in this case study.

We compare the ratio of manually predicted requirements, which are marked as ambiguous with our system-generated result. We calculated that the system detected 60 requirements as ambiguous and 54 as not ambiguous. The proposed approach predicted 39 requirements as ambiguous rightly as per manually tagged requirements as ambiguous. So we computed a precision of 65% and recall of 90%.

C. VALIDITY CRITERIA

Precision and recall are extensively employed evaluation metrics in information retrieval and binary classification applications. They provide insight into the accuracy and completeness of a model or system's performance.

Precision is the degree to which a model or system correctly detects relevant examples. It computes the ratio of accurately predicted positive cases (true positives) to all positive instances anticipated (true positives plus false

TABLE 8. Comparison of CMM results with existing approaches.

Approach		Precision	Recall
Existing approach [2]	$\overline{\sigma}$	57%	65%
	$\sigma_{\min}$	41%	69%
Existing approach [13]	$case3/5$	8.5%	70%
	$case1/5$	9%	41%
Proposed CMM approach	$\overline{\sigma}$	65%	67%
	$\sigma_{\min}$	65%	90%

positives). In other words, precision quantifies the degree of correctness or precision of the model's positive predictions. It determines the ratio of correct positive predictions to the total number of positive predictions and is represented as follows.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Recall, also called sensitivity or true positive rate, shows the capability of an approach to correctly identify all positive cases. It computes the ratio of successfully predicted positive instances, also known as true positives, to all actual positive instances, which are represented by true positives plus false negatives. Recall quantifies the degree to which the model's positive predictions are exhaustive. It determines the proportion of correct positive predictions to the total number of positives, which is depicted below.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

D. PERFORMANCE COMPARISON WITH EXISTING APPROACHES

To find accurate accuracy we performed experiments with the existing approach [2] and the current approach with the same data set and results are given in Table 8. We have also compared the proposed approach with another recent existing approach [13] detecting ambiguity in requirement specification using a knowledge dictionary. The comparison of results given in Table 8 shows the superior performance of the proposed CMM approach.

Please note that $\overline{\sigma}$ calculated using Equation 1 and $\sigma_{\min}$ calculated using Equation 2 represent two different methods to analyze the ambiguity of requirements by comparing similarity with a threshold value. The evaluation of the outcomes is based on the determination of both the mean and minimum precision and recall values. The precision value reflects the accuracy or correctness of positive predictions, reflecting the degree to which the models properly identify important situations. The recall value, on the other hand, shows the comprehensiveness or completeness of positive predictions and measures the models' ability to recognize all real positive events. By evaluating both precision and recall, it is possible to conduct a full evaluation of the performance of the current and existing techniques, thereby gaining insight into the accuracy and completeness with which they capture significant instances. The result shows that the CMM model outperforms the existing work in identifying ambiguities in requirement specifications. Similarly, we have compared our proposed approach with another recently presented existing study [13]. The study presented a method for identifying ambiguity in requirement specifications using a knowledge dictionary constructed by emphasizing transitive verbs and their associated objects in both requirement specifications and supporting documents. Although the existing approach has potential but as compared to the results of our proposed model (CMM) clearly presented better performance as compared to this existing work.

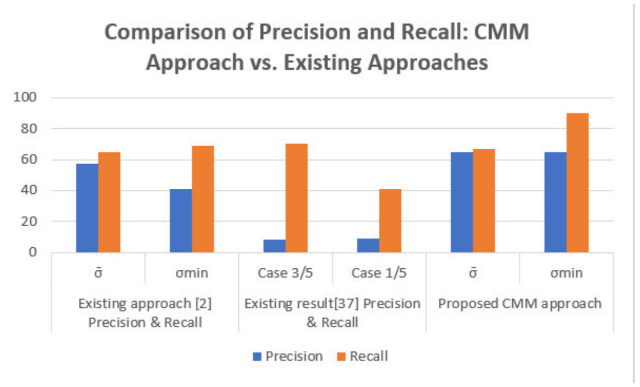


FIGURE 3. Result Comparison between Existing and Current Approach.

Figure 3 provides a visual representation of the comparison derived from the data. The goal of this figure is to show how precision and recall are related, giving a more complete picture of what is shown in Table 8. presents the same results as those found in Table 8. It provides a graphical representation of the results for both the existing approaches and the proposed approach.

V. CONCLUSION AND FUTURE WORK

Pragmatic ambiguity complicates the understanding of natural language requirements and the product cannot meet consumer needs without proper context. Readers interpret requirements differently based on background knowledge which leads to ambiguity. This study addressed this challenge by reducing requirement misinterpretations. The proposed CMM approach is found on graph-based artificial subjects that model the domain background knowledge. After building a domain knowledge graph from domain documents, it uses the shortest path to capture the true relationship among requirements. The shortest path activates highly weighted concepts in a weighted graph to interpret requirements. Activating highly weighted concepts in graphs indicates domain document concept relatedness. Existing methods retrieve concepts from nodes. Based on two neighboring nodes with higher domain knowledge graph occurrences, the shortest path algorithm returns more matching results. Edge-based retrieval of concepts improves requirement interpretation. As per the designed approach, we input a set of requirements and multiple domain documents, where requirements are searched in provided domain documents and receive different interpretations from each domain document. Finally, we find the similarity between retrieved interpretations to decide whether the given set of requirements is ambiguous or unambiguous (i.e., More similarity between retrieved interpretations shows unambiguous requirements and vice versa). We evaluated CMM and observed a precision of 65% and a recall of 90%, surpassing the performance of the existing approach which achieved 51% precision and 63% recall. These findings indicate a significant improvement over current methods and demonstrate the effectiveness of the proposed approach. The current study focuses on reducing pragmatic ambiguity by identifying shortest paths based on nodes and edges. In the future, we aim to achieve better performance with extended paths and auto-selected document input. Additionally, we intend to integrate a larger data set of requirements to investigate the impact of time and requirement quantity.

REFERENCES

[1] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, ''Pragmatic ambiguity detection in natural language requirements,'' in *Proc. IEEE 1st Int. Workshop Artif. Intell. Requirements Eng. (AIRE)*, Aug. 2014, pp. 1–8.

[2] A. Ferrari and S. Gnesi, ''Using collective intelligence to detect pragmatic ambiguities,'' in *Proc. 20th IEEE Int. Requirements Eng. Conf. (RE)*, Sep. 2012, pp. 191–200.

[3] G. Huzooree and V. D. Ramdoo, ''A systematic study on requirement engineering processes and practices in Mauritius,'' *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 5, no. 2, pp. 40–46, 2015.

[4] M. Satpathy, R. Harrison, C. Snook, and M. Butler, ''A comparative study of formal and informal specifications through an industrial case study,'' in *Proc. IEEE/IFIP Workshop Formal Specification Comput. Syst. (FSCBS)*, Apr. 2001, pp. 318–321.

[5] M. Luisa, F. Mariangela, and N. I. Pierluigi, ''Market research for requirements analysis using linguistic tools,'' *Requirements Eng.*, vol. 9, pp. 40–56, 2004.

[6] U. S. Shah and D. C. Jinwala, ''Resolving ambiguities in natural language software requirements: A comprehensive survey,'' *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 5, pp. 1–7, Sep. 2015.

[7] A. K. Massey, R. L. Rutledge, A. I. Antón, and P. P. Swire, ''Identifying and classifying ambiguity for regulatory requirements,'' in *Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE)*, Aug. 2014, pp. 83–92.

[8] M. Bano, ''Addressing the challenges of requirements ambiguity: A review of empirical literature,'' in *Proc. IEEE 5th Int. Workshop Empirical Requirements Eng. (EmpiRE)*, Aug. 2015, pp. 21–24.

[9] B. Gleich, O. Creighton, and L. Kof, ''Ambiguity detection: Towards a tool explaining ambiguity sources,'' in *Proc. 16th Int. Working Conf. Requirements Eng. Foundation Softw. Quality (REFSQ)*, Essen, Germany. Berlin, Germany: Springer, Jun. 2010, pp. 218–232.

[10] A. Ferrari, B. Donati, and S. Gnesi, ''Detecting domain-specific ambiguities: An NLP approach based on Wikipedia crawling and word embeddings,'' in *Proc. IEEE 25th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2017, pp. 393–399.

[11] N. Hussain, H. T. Mirza, F. Iqbal, A. Altaf, A. Shoukat, M. G. Villar, E. S. Flores, M. A. R. Gutiérrez, and I. Ashraf, ''PRUS: Product recommender system based on user specifications and customers reviews,'' *IEEE Access*, vol. 11, pp. 81289–81297, 2023.

[12] M. Dar, F. Iqbal, R. Latif, A. Altaf, and N. S. M. Jamail, ''Policy-based spam detection of tweets dataset,'' *Electronics*, vol. 12, no. 12, p. 2662, Jun. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/12/2662

[13] T. Kato and K. Tsuda, ''A method of ambiguity detection in requirement specifications by using a knowledge dictionary,'' *Proc. Comput. Sci.*, vol. 207, pp. 1482–1489, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922010882

[14] J. Edwards, T. Cassidy, G. de Mel, and T. F. L. Porta, ''Integrating quality of information with pragmatic assistance,'' in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2016, pp. 1–7.

[15] H. Roopa and S. Panneer Arockiaraj, ''The role of artificial neural network in word sense disambiguation (WSD)—A survey,'' in *Rising Threats in Expert Applications and Solutions*. Singapore: Springer, 2022, pp. 221–227.

[16] S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh, and L. C. Briand, ''Using domain-specific corpora for improved handling of ambiguity in requirements,'' in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, May 2021, pp. 1485–1497.

[17] S. Mishra and A. Sharma, ''On the use of word embeddings for identifying domain specific ambiguities in requirements,'' in *Proc. IEEE 27th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 234–240.

[18] S. Kaddoura, R. D. Ahmed, and J. H. D., ''A comprehensive review on Arabic word sense disambiguation for natural language processing applications,'' *WIREs Data Mining Knowl. Discovery*, vol. 12, no. 4, p. e1447, Jul. 2022.

[19] F. Ashfaq and I. S. Bajwa, ''Natural language ambiguity resolution by intelligent semantic annotation of software requirements,'' *Automated Softw. Eng.*, vol. 28, no. 2, pp. 1–45, Nov. 2021.

[20] G. Malik, M. Cevik, D. Parikh, and A. Basar, ''Identifying the requirement conflicts in SRS documents using transformer-based sentence embeddings,'' 2022, *arXiv:2206.13690*.

[21] V. K. C. Manam, J. D. Thomas, and A. J. Quinn, ''TaskLint: Automated detection of ambiguities in task instructions,'' in *Proc. AAAI Conf. Human Comput. Crowdsourcing*, Oct. 2022, vol. 10, no. 1, pp. 160–172.

[22] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, ''Automated handling of anaphoric ambiguity in requirements: A multi-solution study,'' in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE)*, May 2022, pp. 187–199.

[23] A. Yadav, A. Patel, and M. Shah, ''A comprehensive review on resolving ambiguities in natural language processing,'' *AI Open*, vol. 2, pp. 85–92, Jan. 2021.

[24] M. Osama, A. Zaki-Ismail, M. Abdelrazek, J. Grundy, and A. Ibrahim, ''Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements,'' in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2020, pp. 651–661.

[25] S. Saxena, U. Chaurasia, N. Bansal, and P. Daniel, ''Improved unsupervised statistical machine translation via unsupervised word sense disambiguation for a low-resource and Indic languages,'' *IETE J. Res.*, pp. 1–11, Jul. 2022.

[26] R. S. Satpute and A. Agrawal, ''A critical study of pragmatic ambiguity detection in natural language requirements,'' *Int. J. Intell. Syst. Appl. Eng.*, vol. 11, no. 3s, pp. 249–259, 2023.

[27] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios, ''DataTone: Managing ambiguity in natural language interfaces for data visualization,'' in *Proc. 28th Annu. ACM Symp. User Interface Softw. Technol.*, Nov. 2015, pp. 489–500.

[28] B. Kumar, H. B. Maringanti, and K. Asawa, ''Adaptive pragmatic analysis of natural language,'' in *Proc. 1st Int. Conf. Intell. Interact. Technol. Multimedia*, Dec. 2010, pp. 236–240.

[29] A. O. J. Sabriye and W. M. N. W. Zainon, ''An approach for detecting syntax and syntactic ambiguity in software requirement specification,'' *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 8, pp. 2275–2284, 2018.

[30] A. O. J. Sabriye and W. M. N. W. Zainon, ''A framework for detecting ambiguity in software requirement specification,'' in *Proc. 8th Int. Conf. Inf. Technol. (ICIT)*, May 2017, pp. 209–213.

[31] S. Lee, D. Lee, S. Kang, and S.-G. Lee, ''A pragmatic approach to realizing context-aware personal services,'' in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, Dec. 2008, pp. 517–520.

[32] M. Q. Riaz, W. H. Butt, and S. Rehman, ''Automatic detection of ambiguous software requirements: An insight,'' in *Proc. 5th Int. Conf. Inf. Manage. (ICIM)*, Mar. 2019, pp. 1–6.

[33] M. Ramzan, M. Shoaib, A. Altaf, S. Arshad, F. Iqbal, Á. K. Castilla, and I. Ashraf, ''Distributed denial of service attack detection in network traffic using deep learning algorithm,'' *Sensors*, vol. 23, no. 20, p. 8642, Oct. 2023.

[34] K. Ahmed, A. Altaf, N. S. M. Jamail, F. Iqbal, and R. Latif, ''ADAL-NN: Anomaly detection and localization using deep relational learning in distributed systems,'' *Appl. Sci.*, vol. 13, no. 12, p. 7297, Jun. 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/12/7297

[35] F. Iqbal, A. Altaf, Z. Waris, D. G. Aray, M. A. L. Flores, I. D. L. T. Díez, and I. Ashraf, ''Blockchain-modeled Edge-Computing-Based smart home monitoring system with energy usage prediction,'' *Sensors*, vol. 23, no. 11, p. 5263, Jun. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/11/5263

[36] S. Farooq, A. Altaf, F. Iqbal, E. B. Thompson, D. L. R. Vargas, I. D. L. T. Díez, and I. Ashraf, ''Resilience optimization of post-quantum cryptography key encapsulation algorithms,'' *Sensors*, vol. 23, no. 12, p. 5379, Jun. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/12/5379

[37] H. Farooq, A. Altaf, F. Iqbal, J. C. Galán, D. G. Aray, and I. Ashraf, "DrunkChain: Blockchain-based IoT system for preventing drunk driving-related traffic accidents," *Sensors*, vol. 23, no. 12, p. 5388, Jun. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/12/5388

[38] A. Altaf, F. Iqbal, R. Latif, B. M. Yakubu, S. Latif, and H. Samiullah, "A survey of blockchain technology: Architecture, applied domains, platforms, and security threats," *Social Sci. Comput. Rev.*, vol. 41, no. 5, pp. 1941–1962, Oct. 2023, doi: 10.1177/08944393221110148.

[39] X. Li and Z. Ma, "Computational pragmatics: A survey in China and the world," in *Proc. 2nd Int. Conf. Natural Lang. Process. Inf. Retr.*, Sep. 2018, pp. 65–69.

[40] K. A. Mohamed, J. Din, and S. Baharom, "A tool to detect pragmatic ambiguity with possible interpretations suggestion in software requirement specifications," *Int. J. Synergy Eng. Technol.*, vol. 3, no. 2, pp. 52–60, 2022.

[41] A. Ferrari, A. Esuli, and S. Gnesi, "Identification of cross-domain ambiguity with language models," in *Proc. 5th Int. Workshop Artif. Intell. Requirements Eng. (AIRE)*, Aug. 2018, pp. 31–38.

[42] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity and tacit knowledge in requirements elicitation interviews," *Requirements Eng.*, vol. 21, no. 3, pp. 333–355, Sep. 2016.

KHADIJA ASLAM received the master's degree in computer science from the University of Lahore. Currently, she is a Senior Software Quality Assurance Engineer with Strategic Systems International, Pakistan, leveraging more than seven years of practical experience to uphold the quality and dependability of software applications. Her current research interests include testing methodologies and addressing challenges related to achieving clear and unambiguous requirements for requirement specification.

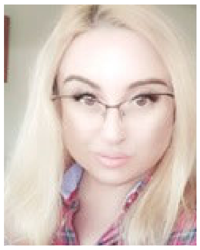
FAIZA IQBAL received the Ph.D. degree from NUST, Pakistan. She directs multiple IoT-based software development initiatives. She is currently with the Department of Computer Science, University of Engineering and Technology, Lahore, Pakistan. Her current research interests include the IoT-based smart applications, knowledge-based systems, network optimization modeling, and data analytics for high-performance protocol design. Professionally, she serves as a member of the Program Committee and the Technical Committee and a reviewer panel for several international journals and conferences. She received the Pakistan's Higher Education Commission Indigenous Scholarship for the M.S. degree leading to the Ph.D. degree.

AYESHA ALTAF received the M.S. and Ph.D. degrees in information security from NUST, Pakistan, in 2009 and 2021, respectively. She is currently an Academician and a Researcher of cyber security. She is also an Assistant Professor with the Department of Computer Science, University of Engineering and Technology Lahore, Lahore, Pakistan. Her professional services include industry consultation, workshops organizer/resource person (workshops/seminars), and a technical program committee member, teaching (U.G./P.G./Ph.D.) courses, research and development, and reviewing for various international journals/conferences. She has almost 15 years of experience teaching at the university level. She has published many scientific research publications in major international journals (ISI-Indexed), such as IEEE Internet of Things Journal, *Journal of Network and Computer Applications* (Elsevier), *Journal of Systems Architecture* (Elsevier), IEEE Access, and *Computers and Electrical Engineering* (Elsevier), with a cumulative IF of more than 60.

NAVEED HUSSAIN received the Ph.D. degree in computer science from Comsats University Islamabad, Lahore Campus, Pakistan. He is currently an Assistant Professor with the Department of Software Engineering, University of Central Punjab, Pakistan. He has published several articles in reputed journals. His research interests include data mining, sentimental analysis, machine learning, and opinion mining.

MÓNICA GRACIA VILLAR is currently with Universidad Europea del Atlántico, Santander, Spain. He is also affiliated with Universidade Internacional do Cuanza, Kuito, Bié, Angola, and Fundación Universitaria Internacional de Colombia, Bogotá, Colombia.

EMMANUEL SORIANO FLORES received the bachelor's degree (Hons.) in business administration, the master's degree in financial management, the master's degree in corporate business communication, the master's degree in China–Asia Pacific business, the master's degree in international business administration, the master's degree in educational innovation, and the Ph.D. degree in higher education. He has experience as a Professor and a Researcher at various universities in Mexico and Spain. He is currently the Coordinator of the Master in Business Administration with the European University of the Atlantic, Spain.



ISABEL DE LA TORRE DÍEZ is currently a Professor with the Department of Signal Theory and Communications and Telematic Engineering, University of Valladolid, Spain, where she is also the Leader of the GTe Research Group (http://sigte.tel.uva.es). Her research interests include the design, development, and evaluation of telemedicine applications, services and systems, e-health, m-health, electronic health records (EHRs), EHRs standards, biosensors, cloud and fog computing, data mining, quality of service (QoS), and quality of experience (QoE) applied to the health field.

IMRAN ASHRAF received the M.S. degree (Hons.) in computer science from the Blekinge Institute of Technology, Karlskrona, Sweden, in 2010, and the Ph.D. degree in information and communication engineering from Yeungnam University, South Korea, in 2018. He was a Postdoctoral Fellow with Yeungnam University, Gyeongsan, South Korea, where he is currently an Assistant Professor with the Information and Communication Engineering Department. His research interests include positioning using next-generation networks, communication in 5G and beyond, location-based services in wireless communication, smart sensors (LIDAR) for smart cars, and data analytics.