Semantic Interoperability Through a Novel Cross-Context Tabular Document Representation Approach for Smart Cities

SHUO YANG¹, (Member, IEEE), AND RAN WEI², (Member, IEEE)
¹School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China
²Department of Computer Science, University of California, Irvine, Irvine, CA 92697, USA
Corresponding author: Shuo Yang (yangshuo@gzhu.edu.cn)
This work was supported in part by the National Natural Science Foundation of China under Grant 61802079, and in part by the Guangzhou University under Grant RP2020125.

ABSTRACT Semantic interoperability is the process of representing, editing, and transmitting semantic information in one context, and then receiving and interpreting it in another. It is an important research topic in semantic web, Internet of things, smart city, smart enterprise and artificial intelligence. The current research methods of semantic interoperability generally include standardization, ontology modeling and collaboration templates, each of which has its own internalized limitations. In this paper, a novel cross-context tabular document representation approach (Tabdoc) is proposed, where semantic documents used for cross-domain interaction can maintain the same syntax, concepts, and semantic relations. The new method is implemented under Sign Description Framework (SDF) and an editor is newly developed for semantic document creation. Finally, theoretical analysis and experimental evaluation on open datasets demonstrate the feasibility and efficiency of the proposed scheme. Experimental results indicate that Tabdoc approach can effectively identify, extract and consistently interoperate semantic components of tabular documents.

INDEX TERMS Data integration, information exchange, information retrieval, inference mechanisms, semantic interoperability.

I. INTRODUCTION
Among all the sophisticated technologies used to realize smart cities, information integration has attracted more and more attention [1], [2]. In recent years, information integration and interoperability have become important research topics in database, information system [3], [4], industrial automation [5], semantic web [6], Internet of things [7], [8], pervasive computing [9] and so on. In the real world, much knowledge is acquired from documents including technical reports, journals and magazines, to name a few [10]. Therefore, the research on document interoperability become an important technique to implement information integration [11]–[13]. It has wide applications in the interoperation between ERP systems, clinical systems [14], e-business systems [15], etc. However, documents not only contain syntactic schema but semantic meaning (or features). The interoperation of documents with different syntax and semantic representations is easy to say than done.

In 1998, the notion of action sheets is proposed [16]. It is one of the most important steps in achieving document interoperability, allowing event-based behavior to be separated from document structure and presentation characteristics [16]. This concept is particularly useful for XML. In XML documents, external stylesheet rules can associate any presentation characteristics with specific XML elements [16]. Similarly, external action rules can also associate any event processing mechanisms with specific elements of an XML document [16]. However, in most cases, different users may design different DTDs; therefore, the same action sheet cannot be reused for elements across multiple documents from different users. The root cause is that the action sheet does not consider the processing of documents at a semantic level. In practice, people often represent and edit a document in one context, and then receive and interpret it in another. For example, when heterogeneous industrial information systems automatically inquire about products, make a
valid offer, negotiate on industrial terms, and sign a contract on behalf of the parties involved, the documents exchanged between them need to be semantically interpretable and interoperable. In this paper, this type of documents is called semantic document. Specifically, a semantic document is a document that is represented by humans and computers in a readable and understandable form [17]. Cross-context semantic document interaction has attracted much attention in semantic web [18], e-business [19] and artificial intelligence [20]–[22]. It can also serve as the foundation of service collaboration in Internet of things (IoTs) [12].

Current researches on semantic document interaction (or interoperability) is mainly divided into three categories: standardization [23], ontology modeling [24] and collaboration templates [25]. These methods have their own merits and drawbacks. Specifically, standardization is adapted to create uniform documents. However, the document types and elements created are rigidly limited to pre-defined standardized types [26], [27]. Therefore, it is difficult to construct complex documents without constraints by utilizing standardization methods. To compensate this disadvantage, ontology modeling method [28], [29] is proposed, which can design most of complex documents when ontologies are carefully designed. However, the limitation of this approach is the “domain-wide” problem. It means a semantic document created under one ontology context (or domain) may not be accurately interpreted by another context (or domain) [17]. Furthermore, another disadvantage shared by the standardization and ontology modeling approaches is that they cannot guarantee semantic consistency of instances filled in a document template transferred between document sending and receiving parties. To resolve these problems, collaborative template approach [17], [25] [30], [31] [32] is proposed, which well connects heterogeneous domains by collaboratively building semantic documents. However, its drawback is that every document template exchanged must be collaboratively created first [17]. Document types that have not been collaboratively created cannot be semantically interoperated between different domains or contexts [17]. Based on the discussion above, two research problems are summarized as below:

Problem 1 (Document Complexity Problem): Semantic documents are complex because each type of document is a complex semantic phenomenon. Users should be allowed to have autonomy, that is, create any semantic document based on the user’s own ideas and requirements. However, due to the flexibility of language grammar and the complexity of conceptual relationships between terms in semantic documents, it is not easy to design, create, parse and use complex semantic documents.

Problem 2 (Interoperability Problem or Cross-Context Problem): Semantic documents of different domains have interoperability problems in three levels: document syntax (i.e., few users utilize the same document representation grammar), document relation (i.e., users often represent a semantic relation in their own ways), and document vocabulary (i.e., document users are autonomous to instantiate the same document template by using different vocabularies). The cross-context problem hinders a semantic document correctly composed by a document writer in one place from being consistently processable and understandable by a document reader in another place [33]. However, many traditional methods (e.g., [17], [25]) only partially consider limited structural relationships (e.g., ancestor-descendent relationships or sibling relationships among XML nodes), which remains insufficient for efficient cross-context semantic disambiguation [34].

To resolve the above two problems, this paper proposes a Tabular Document Representation (TabDoc) method, which represents heterogeneous semantic documents across domains in a consistent way; thereby making documents in various contexts interoperable at both the syntactic and semantic levels. The central idea of this approach is to adopt a “divide and conquer” strategy by defining complex semantic documents as a three-level structure of vocabulary, relationships and documents. Through this division, complex interoperability problems are divided into independent sub-problems of concept interoperability, relationship interoperability and document interoperability. The main contributions of this paper include:

- It implements information interoperability through a novel semantic document representation approach (Tabdoc) which allows consistent syntactic processing and semantic understanding between semantic documents sending and receiving parties.
- It contributes to the intelligence of document interpretation and processing through a novel semantic interpretation algorithm, which is effectively adapted to changing business conditions and contexts.

The paper is organized as follows: Section II proposes a novel Tabdoc approach to resolve problems of document complexity and interoperability. Section III develops a Tabdoc Editor to implement Tabdoc approach with an example to show its application. In Section IV, experimental evaluations on open datasets demonstrate the feasibility and efficiency of the proposed scheme. Section V theoretically compares the proposed approach with related ones. Finally, conclusion of the paper and future works are given.

II. METHOD: A NOVEL TABDOC APPROACH

This section proposes a novel tabular document representation (Tabdoc) approach to resolving the two problems so as to reduce document complexity and achieve semantic interoperability between exchanged semantic documents across contexts.

A. OVERVIEW

Tabdoc approach (shown in Fig. 1) is a holistic solution for how to implement cross-context semantic document

1. cXML. Available: http://www.cxml.org/.
2. ebXML. Available: http://www.ebxml.org/.
interoperation (or interaction). It is divided into three levels: vocabulary, relation and document.

The vocabulary level aims to solve the problem of conceptual interoperability by using the general CONEX dictionary (CoDic) [30], [35]. CoDic is a common vocabulary designed by parties involved in information exchange under the CONEX project [31]. Any term in CoDic dictionary is uniquely identified by an internal identifier (iid ∈ IID) [31]. This identifier is neutral and independent of any natural language [31]. In CoDic, there are two types of concepts: (1) common concepts: accurately used by different industrial or business groups in different natural languages, i.e., concepts that are shared by common vocabularies (CV) and (2) local concepts: uniquely used by individual groups, i.e., concepts that are unique in local vocabulary (LV), local document templates, and locally instantiated documents. CoDic guarantees that all terms are accurate and semantically consistent without ambiguity in the concept level.

The relation (or concept type) level uses a new complexity conceptualization strategy (see Section II-C) to resolve the document complexity problem. It views any semantic document as groups of compound concepts (Section II-C), each of which consists of a set of atomic concepts concatenated with semantic relations (Section II-C). Each atomic concept represents a single sememe, while compound concept consists of several sememes with their inter-relations. Thus, it is necessary to summarize as many compound concept types as possible and illustrate how each of them is represented in semantic documents. The more compound concept types are represented, the more the variety of the document content can be expressed.

The document level achieves the document interoperability among different contexts based on a newly proposed de-contextualization strategy. It includes syntactic de-contextualization and semantic de-contextualization. The reason for designing syntactic de-contextualization is because the grammar rules for document design and representation are context-sensitive and not interoperable between different contexts. It is very common for people from different environments to use different document representation grammars or syntaxes. For example, some people are required to design document syntax in XML, while others prefer designing it in JSON, and different people follow different rules of document representation. For example, Xiao et al. [17], [36] proposed the method of document design using triplet format. This prevents the document from being syntactically interoperable between different domains (or contexts). To solve this problem, a syntactic de-contextualization for syntactical neutralization is needed to make a document be syntactically interoperable. In addition, the lack of semantic interoperability is obvious for online message exchange, because many web services have proprietary interfaces and were originally devised for standalone applications that may only handle with particular kinds of input documents [11], [12]. To handle this problem, semantic de-contextualization is proposed to enable documents understood unambiguously in heterogeneous contexts.

In summary, the vocabulary level and concept type level provide common concepts and relations, which serves as the foundation to fulfill cross-context semantic understanding. The document type level takes responsibility for cross-context consistent document interoperation by using a novel de-contextualization strategy.

B. SYNTACTIC DE-CONTEXTUALIZATION

Syntax is the set of rules, principles, and processes that govern the structure of statements in a given language. To realize syntactic de-contextualization, we design and implement a universal and scalable document representation grammar. It aims to construct general grammatical rules on organization of terms for reading, understanding and editing documents. It needs to conform to two principles. First, the universal document syntax should follow a context-free grammar. Normally, a context-free grammar is a four-tuple consisting of a set of non-terminal variables, terminal variables, production rules and a starting variable [37]. The universal document syntax, acting as a context-free grammar, is the foundation of building a universal document parser, and thus it needs to follow particular rules for easy parsing. Second, the language generated from the universal document syntax should be a collaborative language. This means it allows language users to collaboratively update the language based on their practical demand.

Tabdoc Grammar, as a universal document syntax, has two types of productions:

\[
sign ::= \text{sign} (\text{sign} [\text{Attlist}]^{*})^{*} | \varepsilon \quad (\text{II.1})
\]

\[
\text{Attlist} ::= \text{attribute} – \text{value pairs} \quad (\text{II.2})
\]
where the “sign” in the equation (II.1) is a nonterminal. The symbol ε is indicated as empty. The “Attslist” in equation (II.2) is a set of properties of any sign element in Tabdoc documents. Tabdoc language is a sequence of strings that are derived from the Tabdoc syntax by executing production rules. In every derivation step, one or several productions (or rules) are used to derive the non-terminal. This section designs Tabdoc language, also known as Tabdoc schema, to construct document templates and document instances. The entire process of document creation is described in equation (II.3).

\[ \text{TabGram} \quad \Rightarrow \quad \text{Tabdoc language (Tabdoc schema)} \quad \Rightarrow \quad \text{document template} \quad \Rightarrow \quad \text{document instance} \quad \text{(II.3)} \]

where Tabdoc Grammar (TabGram) is equivalent to language rules that govern various sentence components (e.g., subject (S), predicate (V), and object (O)). It can produce syntactic structures such as “S+V+O” or “S+V+ADD1+O”. Such a series of syntactic structures can be combined to produce a schema, Tabdoc language (or Tabdoc schema), which is similar to XML schema. In Tabdoc language, each syntactic component is represented by a simple symbol \( \text{sign} \). After adding attributes to each sign, we can get a document template. Then, populated each attribute with values, a document instance is formulated. Currently, Tabdoc schema is implemented in XML schema [38], but it can be developed by other methods (e.g., JSON).

1) DOCUMENT REPRESENTATION UNDER TABDOC

Syntactically, document design is about how to organize terms to create a well-formed and grammatically valid pattern. In Tabdoc syntax, semantic documents take the form of nested tables, which can be formalized as a list of ordered matrices arranged in a hierarchy. These hierarchical nested tables form a matrix tree structure, in which each table in the hierarchy is described as a matrix and built in order. The matrix tree (MTree) \( d_k \) is defined in (II.4).

\[
d_k = \begin{bmatrix}
c_{r_1c_1} & \cdots & c_{r_1c_m} \\
\vdots & \ddots & \vdots \\
c_{r_ic_j} & \cdots & c_{r_mc_m} \\
\end{bmatrix} \quad \text{(II.4)}
\]

where \( c_{r_ic_j} \) is a cell in the \( i \)th row and the \( j \)th column (\( 0 < i \leq n, 0 < j \leq m \)) of a table. A matrix tree can also be notated in (II.5).

\[
d_k = M_{m_0n_0}^0(1,1).M_{m_1n_1}^1(I_1,J_1)\ldots M_{m_kn_k}^k(I_k^1,J_k^1)| (I_k^2,J_k^2)|\ldots). (M^{k+1}|\ldots) \quad \text{(II.5)}
\]

where \( M_{m_0n_0}^0(1,1) \) is a root cell. Any matrix \( M_{m_kn_k}^k(I_k,J_k) \) represents an embedded table in the \( k \)th level with dimension \((m_k \times n_k)\) in the hierarchical structure of MTree (see Fig. 2), which can contain a sub-table in its cell \((I_k,J_k)\) or sub-tables (e.g., separated by vertical lines in (II.5)). The location of a cell \( C_{ik}(i,j) \) is called the cell identifier (cid), where the cells of each table are represented in a sequential structure. Equation (II.6) uniquely identifies the cid of a cell:

\[
cid = (I,J)_1.(I,J)_2\ldots(i,j)_k \\
= 1.(I_1,J_1),(I_2,J_2)\ldots(i_k,j_k) \quad \text{(II.6)}
\]

In order to access the cells in the \( i \)th row and the \( j \)th column in the \( k \)th level, it is only necessary to traverse those cells in the previous \((k-1)\) levels matrices that have the nested matrices. \((i_k,j_k)\) denotes the position of the cell nested in the innermost sub-table in the hierarchical MTree. \((I_k,J_k)\) refers to the coordinate position of a cell in the up-level nesting table at the \( x \)th level, and the cell is used to nest the \((x+1)\)th level matrix. Figure 2 shows an example of the MTree structure of a Tabdoc document.

2) ATTRIBUTE DESCRIPTION FOR CELLS

In actual document use, because the description of cell characteristics in the document is complex, multiple attributes need to be defined to express them. This section divides the attribute set of the cell into five categories, as shown in Table 1. These categories are used to represent cells in five dimensions: identification, data model, subcell, instance, and presentation style. Formula (II.7) describes a set of attributes (Atts) in any cell in a Tabdoc document (or a sign \( \sigma \) in Tabdoc schema).

\[
\text{Atts} ::= \text{ref, pid, term, size, st, max, min, pos, cho, sel, imp, pri, inst, ifof, dt, op, f, r, disp, style} \quad \text{(II.7)}
\]

where PID and REF are used to form a sign \( \sigma \). PID indicates the form of \( \sigma \), and REF represents the intrinsic meaning and
TABLE 1. Attributes of a cell or a sign (see Fig. 3).

| Attribute (feature) | Name             | Definition                                                                 | Description |
|---------------------|------------------|----------------------------------------------------------------------------|-------------|
| (1) The location and meaning of a cell |                      |                                                                            |             |
| Template identifier | tid = uid + tid_no | Unique template id of a user                                           |             |
| Instance identifier | iid = tid + iid_no | Unique instance id of a user                                          |             |
| Position identification | pid = cid      | Location of the current cell (or sign)                                 |             |
| Reference (mandatory) | ref = iid_1,...,iid_n ∈ CoDic or ref = iid ↓ dt ∈ CoDic | Define meaning of current cell in pid by referring to iids ∈ CoDic or an iid-ed data type |             |
| Term (mandatory if ref = dt, else optional) | term : literal ← iid of dt | Term if σ is a data type and specific literal term in local language must be provided |             |
| (2) The sign type of a cell and the associated subcells |                      |                                                                            |             |
| Sign type (or data model) | st = cell + col + row for current cell | Define sign type such as table, row, column or cell |             |
| Size of a compound sign (or a cell) | size = i | j | (i, j) | Define the size of a sign (or a cell) |             |
| Cell position (mandatory if st=table) | cid (or pos) = i | j | (i, j) | i represents the row number, j represents the column number |             |
| (3) Constraints on subcells of a cell (optional) |                      |                                                                            |             |
| Occurrence | min ∈ (0, n) | Minimum number of subcells |             |
| Sequence | max ∈ (0, ∞) | Maximum number of subcells |             |
| Choice | cho = 1 | n (current cell); sce = yes | no (subcell) | Single or multiple choice |             |
| Priority/Importance | Imp = yes | no (current cell); pri = n (subcell) | Describe the importance and priority order of subcells |             |
| Semantic relation | sem (or rt) = (types of semantic relations) | Reserved attribute for relation definition |             |
| Anchor | anc = [pos*] | Pointing to related cells |             |
| (4) Constraints on instance of a cell (optional) |                      |                                                                            |             |
| Value type (omit if not allowed) | st = val | Value type is a special instance of st |             |
| Instance | inst = value or values | Instance of a cell or sign |             |
| Instance of | iid = cid | Denote current sign as an instance of another sign in cid |             |
| Data type | dt = data types defined in XML and Tabdoc | Data types constrain how to instantiate the value of the current sign |             |
| Operation | op = is empty | [lt | gt | lte | gte | lte | gte | likely | not | reciprocal | factorial | ceiling | floor | …] | An operation represents action, state, willingness, possibility, and judgement of a sign and constrains its instance value |             |
| Formula | f = math formula | Mathematical formula |             |
| Rule | r = if x_1,...,x_n then y or a reference of a rule | A well-formed rule deriving instance of current cell |             |
| Linguistic grammar | lg = [S+V+O | P | …] | A reserved attribute to constrain language grammar |             |
| (5) Constraint on the presentation of a sign |                      |                                                                            |             |
| Display or presentation | disp (or pr) = inst | term | ref | How to display current cell |             |
| Style | style = css | How a sign is presented in a particular css style |             |

* This is an updated version from previous releases. Please refer to [39] for more details.

**Grammar**

The denotation of σ, which follows the dual symbol model theory of Saussure [39]. The other attributes are the modifiers for σ. Specifically, attributes such as ST, POS, MIN, MAX, CHO, SEL, IMP, and PRI are employed to describe the functionality and connotation of σ [40]. Attributes such as DT, OP, F, and R are utilized to instantiate an abstract σ in the form of “sign σ[pid, ref, op, dt, f, r] < iid or literal >” [40]. For example, the statement “fridge is white” can be expressed as: σ[pid = “(1, 3),(1, 2)”, ref = “5107df022918”, op = “is”] < 5107df022918 >, where in the CoDic dictionary, “iid = 5107df022918” means “refrigerator”, and “iid = 5107df02f309” means “white” [35].

3) IMPLEMENTATION OF TABDOC GRAMMAR

To concretely create and edit a semantic document, Tabdoc Grammar is implemented under Sign Description Framework (SDF) [35] in XML. In this paper, the implemented Tabdoc

Grammar is called SDF Tabdoc which aims to be compatible with most existing XML-based document processors. SDF Tabdoc has only one XML element, called < sign > that defines a concept with a set of attributes shown in Table 1. < sign > is nested to construct Tabdoc schema which will be populated with attributes for Tabdoc document templates and then filled with values for Tabdoc document instances. Fig. 3 presents the XML schema for SDF Tabdoc, which is simple, universal and context-free.
C. SEMANTIC DE-CONTEXTUALIZATION

Semantics is the language meaning assigned to a term, phrase, sentence, paragraph and even an article. Document semantics includes meaningful terms and semantic relations between terms [17]. Semantic relations include the explicit relation (e.g., instance relation) and implicit relation (e.g., linguistic grammar) between lexical terms of a document. Thus, consistent understanding of a document needs to guarantee the accurate interpretation of the concepts of all terms and their relations without the interference in different contexts. Thus, semantic de-contextualization method is proposed. The idea is simple: each concept of a term is semantically assigned with a unique iid referring to the unique meaning of the term in CONEX Dictionary (CoDic) that follows the CONEX collaboration principle [30], [35]. Each type of semantic relation also semantically corresponds to a unique iid referring to a unique meaning of the relation in CoDic. Thus, the semantics of any document designed is context-free because it has been transformed to a set of hierarchically sequence of iids.

1) COMPLEXITY CONCEPTUALIZATION STRATEGY

In Tabdoc, any complex semantic document is tabularized in an array of rows and columns. Each simple cell is only filled with a single concept which is called atomic concept (see Definition 1). Several simple cells are hierarchically combined together to represent a complex concept which is called compound concept (see Definition 2). A semantic document can be partitioned into several independent semantic units for easy interpretation. Each semantic unit can be conceptualized as a compound concept with a certain type or several compound concepts combined with more complex semantic relations.

Definition 1 (Atomic Concepts, AC): An atomic concept is a collaborative sign (cosign) with atomic meaning, which is defined as:

\[ AC = (S, W, D, L). \]  

(II.8)

where W is any character string with a common identifier S and definition D. L is a natural language describing W and D. Atomic concepts are collaboratively built by experts from different domains. For example, (0 × 5107d00d93, “employee”, “contributes labor and/or expertise to endeavor of an employer and is usually hired to perform specific duties which are packaged into a job”, “English”) is an atomic concept.

Definition 2 (Compound Concepts, CC): A compound concept defines an association or relationship among a set of atomic concepts (AC), which is defined as:

\[ CC = (GID, AC_{i_1}, \ldots, AC_{i_m}, CCT, AC_{j_1}, \ldots, AC_{j_n}). \]  

(II.9)

where any compound concept CC with type CCT is identified by a group identifier GID; AC_{i_1}, \ldots, AC_{i_m} is a sponsor of CCT, while AC_{j_1}, \ldots, AC_{j_n} is a consumer. For example, cc1=(001,invoice, part-of, product, price, tax, buyer, seller) means that an invoice has product, price, tax, buyer and seller as its components. In applications, CCT (e.g., part-of in the example) is replaced by its identifier.

Compound concept types (CCT) represent the relationship between atomic concepts, which is defined as:

\[ CCT = (S_{cct}, D_{cct}) \]  

(II.10)

where \( S_{cct} \) is a common semantic relation identifier for a compound concept type; \( D_{cct} \) is a definition of \( S_{cct} \). The more compound concept types are defined, the more complex semantic documents can be represented. There are many types of relationships between concepts in real natural languages. Therefore, compound concept types are diverse and difficult to be listed thoroughly. This paper mainly uses eight types of compound concepts borrowed from the field of information science [41] for Tabdoc document representation. They are reference relation, part-of relation, parallel relation, calculation relation, sequence relation, progressive relation, choice relation and instance relation, whose definitions can be found in [42].

Users can easily create a document template under Tabdoc Grammar with a newly designed tabular editor for the expression of mind-thinking, concept selection, and relationship building. Specific steps are as follows.

Step 1: Create a cell on the computer screen as the root cell of the row, column, or table, and fill in the value of the property list in that cell.

Step 2: Expand the root cell to a row, a column, or a table.

Step 3: Traverse each empty cell and specify the value of its attribute list to define them and establish a semantic relationship (i.e., CCT) between multiple cells.

Step 4: Repeat step 3 until users no longer need to specify attributes for creating sub-tables, sub-rows, sub-columns, or sub-cells.

When specifying attribute values for each cell, a document user applies the context-free common terms \( \tau \subset CoDic \) [30], [35] as the semantic references of all signs \( \sigma \), such that a meaning \( m : (iid \leftarrow \tau) = (iid \leftarrow \sigma) \), that is, the iid of each \( \sigma \) in the Tabdoc refers to the iid of the common concept in CoDic. Since each term (or sign) meaning in a document template is unique and context-free in the form of an “iid”, the document template (i.e., relations between multiple signs) is also unique and context-free. This allows any document template to be personal to an individual user but still semantically interoperable for external document receivers.

2) SEMANTIC INTERPRETATION

This paper uses inference on rules for semantic interpretation, which means it uses rules to capture the semantics of a document. Thus, rule execution is the process of understanding of a semantic document. Using rules as a semantic carrier has two benefits. The first is the freedom in rule design and creation due to the generality of the context-free rule syntax [43], [44]. Second, from a technical perspective, user-defined semantic relation types and attributes can be
automatically transformed into patterns of rules. These rules can participate inference for semantic interpretation without requiring new function definition for semantic parsing to cope with different contexts. Users only need to declare transformation paradigms to define how they are transformed into rule patterns. For example, in Drools, a "\.dsl" file can be transformed into rule patterns via a corresponding "\.dsla" file. The semantics of rules is also context-free due to the direct inheritance from the IID-based context-free Tabdoc document.

Stage a) Logical structure extraction of a Tabdoc document

In the first stage of semantic interpretation, a Tabdoc document is logically understood via a Vector Tree Model (see Definition 3). The logical structure is the hierarchical conceptual relation between semantic units in a Tabdoc document.

Definition 3 (Vector Tree, VTree): Any document that can be tabularized is further logically represented by a vector tree structure $VT^3$:

$$VT = (I_1^1, I_1^2, \ldots, I_k^1, \ldots, I_m^n),$$

(II.11)

where (1) each node in $VT$ is represented by $I_i^j$; (2) the level of a node in $VT$ is $q \in (1, \ldots, k, \ldots, n)$; (3) sibling nodes are of the same level and represented by $p \in (0, \ldots, i, \ldots, m)$; (4) the parent of a node at level $k$ is represented by a vector $(I_1^1, \ldots, I_k^{k-1})$; (5) the children of a node at level $k$ is a set of vectors $(I_1^1, \ldots, I_k^{k+1})$ and (6) root of $VT$ is a one-dimensional vector $I_1^1$ with $q = 1$ and $p = 1$.

Each node in a Vector tree (VTree) represents a concept in a Tabdoc document. The position of a node in VTree refers to its corresponding cell identifier, which ensures that VTree is mappable with the Matrix Tree (MTree). Based on different compound concepts defined in the complexity conceptualization strategy (Section II-C), a Vector Tree (VTree) Building Algorithm is proposed to construct the logical structure of a Tabdoc document. VTree building algorithm is designed to extract semantic components including all entities, values and relations in the document and construct a tree structure for visualization. The steps of VTree building algorithm are as below:

Step1: Based on atomic concepts and the semantic relations between them, basic sub-VTrees (e.g., Fig. 4) are first constructed.

Step2: Based on the semantic relations between compound concepts and sub-VTrees obtained from Step 1, complex sub-VTrees are constructed.

Step3: Repeat Step 2 until all compound concepts are extracted as independent semantic units.

Step4: Combine all complex sub-VTrees to form a complete VTree by defining a shared root node.

Stage b) Semantic interpretation algorithm (SIA)

This section proposes a novel semantic interpretation algorithm (SIA) for Tabdoc document understanding based on an improved Rete algorithm (IRA). IRA builds a network $<N, E>$ where $N$ is a set of nodes and $E$ is a set of edges for node linking. Each node represents one or more tests in the predecessor (LHS) of a rule, and each node has one or two inputs and any number of outputs. The network will handle the facts that are added to or deleted from the working memory. The input and output nodes are located at the top and bottom of the network, respectively. The various types of nodes described above together form a Rete network, and the network is the way in which the improved Rete working memory works. The pseudo-codes of SIA and IRA are shown in Table 2 and Table 3, respectively.

The novel SIA consists of several steps as follows:

Step 1: Based on the logical structure constructed by VTree Building Algorithm, it is easy to declare the fact templates (FT) of a Tabdoc document. This step is similar to object-oriented programming of defining which concept should be a class name and which concepts should be its attributes.

Step 2-3: Based on the FT extracted in Step 1 and the instantiated content extracted from Tabdoc document (Doc) by the VTree building algorithm, facts are created and inserted into the working memory (WMDoc) that serves as a database to store facts transformed from a Tabdoc document. In addition, some facts can be pre-defined and stored in the database ready for inference in document receiver side.

Step 4: This step aims for rule creation. According to the source and functionality, rules have three categories. The first category is RuleDoc(DDoc) created by Tabdoc documents. This kind of rule has three sub-types: (1) rules for querying information in a Tabdoc document; (2) rules for creating facts (e.g.,

![Figure 4. Sub-VTree structures of CCTs. SCIR is short for single column instance relation and MCIR is short for multiple columns instance relation [42].](image)

### TABLE 2. Semantic Interpretation Algorithm (SIA)

| Semantic Interpretation Algorithm (SIA) |
|-----------------------------------------|
| **Input:** Tabdoc document (Doc)        |
| **Output:** Tabdoc parsing results      |
| 1) Fact template (FT) declaration;      |
| 2) Facts ($F_{Doc}$) creation based on  |
| FT and instances extracted;            |
| 3) Insert $F_{Doc} \rightarrow WMDoc$;  |
| 4) Load rules $R_{Doc}$ and rules $R_{DB}$ into product memory (PM); |
| 5) Based on $WMDoc$ and PM, create knowledge session (KS) that continuously communicates with rule engine (RE) by loading new facts and rules; |
| 6) Run IRA algorithm to activate rules with facts by pattern matching; |
| 7) Execute each activated rules by agenda following conflict resolution strategies. |

3 More detailed description about VTree can refer to [43]
TABLE 3. Improved Rete Algorithm (IRA).

| Improved Rete Algorithm (Rete improved) |
|----------------------------------------|
| **Input:** productions (rules)        |
| **Output:** Rete network (combination of Alpha net and Beta net) |
| 1) Create a root node (root);          |
| 2) Add a rule (r)                      |
|   a. Read a pattern (p) from r (i.e., an att-val pair in the LHS), and then check the type of p (e.g., predicate); |
|   b. Check whether the corresponding alpha node α exists for p. |
|     If α exists:                       |
|       Record the location of α;       |
|     else:                             |
|       Add p as a new α to the network and create a memory table (MT) of α based on p; |
| 3) Repeat step (b) until all patterns (P, p ∈ P) have been processed; |
| 4) Create Beta nodes (β);             |
|   Left input node to β₂ is α₁;        |
|   Right input node to β₂ is α₂;       |
|   Left input node to βᵢ is βᵢ₋₁;      |
|   Right input node to βᵢ is αᵢ, (≥2); |
|   Inline MT of αᵢ and MT of βᵢ₋₁; to create MT of βᵢ; |
| 5) Repeat step (d) until all Beta nodes have been processed; |
| 6) Action part is encapsulated as a leaf node (or Action node) and taken as the output node of βᵢ; |
| 7) Repeat step 2) until all rules are processed. |

a user-filled single choice); (3) rules for computing arithmetic results of mathematical formulas. The second category is RuleExe (RExe), which is used for program execution. This means the document parser program is also represented in the form of rules. In this way, document processing does not mean the document parser program is also represented in a human readable form. In this way, document processing does not mean the document parser program is also represented in a human readable form. RuleExe also includes three sub-types: (1) rules for loading facts parsed from Tabdoc document into working memory; (2) rules for assessing which facts do not meet the limitations and requirements in the Tabdoc document; (3) rules for loading the priority of different rules. The third category is business rules and processes, which is beyond the discussion of this paper.

Step 5-7: Load the facts in working memory and rules in production memory into pattern matcher and run IRA algorithm to execute pattern matching between facts and the condition parts (LHS, left-hand side) of rules. Each matched rule will be marked as an activation status, which will be waiting for execution of the header parts (RHS, right-hand side) of the rules. The execution procedure is controlled by an agenda that takes responsibility of resolving conflicts by using various conflict resolution strategies (e.g., priority), which is also out of the research scope of this paper.

III. IMPLEMENTATION: TABDOC EDITOR
This section designs a tabular document editor, Tabdoc Editor, to implement Tabdoc approach. By the Tabdoc Editor, the semantics of any document are context-free because it has been transformed to a set of hierarchically sequences of iids.

A. SYSTEM PROTOTYPE: TABDOC EDITOR
Tabdoc Editor can implement Tabdoc documents under Tabdoc Grammar and parse its semantics via a generalized inference engine. Tabdoc Editor consists of components of Table Console, Property List, Tabdoc Parser, Semantic Input Method (SIM) [17] and CONEX dictionary (CoDic) [17] for document creation and components of Inference Engine (including VTree building, SIA and IRA algorithms) for semantic interpretation.

Table Console creates tables (including cells, rows and columns) based on the property values and instance values entered by document template designers and users. Property List provides both template designers and users a form to define and operate each cell of a tabular document. Based on a Tabdoc template, a template user can instantiate it into a Tabdoc instance. Tabdoc template is checked for its well-formedness under Tabdoc schema by Tabdoc Parser. Tabdoc instance is also checked for its validity when instantiating a Tabdoc template. In Tabdoc Editor, template designers and users should adopt SIM to input signs from CoDic. All inputted words are restricted to CoDic unless the inputs are literals.

After a Tabdoc document is received, the novel vector tree (VTree) building algorithm is used for the construction of logical structure through the properties defined during the document creation. By the preorder traversal of the tree structure, we analyze the type of each node through the sign type property (st) to distinguish whether it is an entry, or a label, or a semantic relation type. When we read a label (i.e., property st=cell), we can extract its identifier value from its property ref (i.e., reference) and then search in the CoDic dictionary to get its concept. Its corresponding value cell can be identified based on the logical structure through the positional property iof (i.e., instance of) and the instance property ins (i.e., instance value). The semantic relation between them can be acquired from the property sem (or op). Thus, when we read the sub-tree structure of a semantic relational type, we can construct the semantic relation components, such as “entry-label pair” or “label-label pairs”.

B. EXAMPLE OF TABDOC EDITOR USAGE
This section takes a ticket booking scenario as an example to show how the Tabdoc Editor implements cross-context document understanding. In this example, the booking party writes the document in Chinese and send it to the party who can only understand English. The Tabdoc approach addresses the problem of heterogeneous semantic interoperability by identifying any concept as a context-free iid. Specifically, this example implements the solution through the semantic transformation of “Chinese instance → context-free iid-based instance → English instance”. With this semantic transformation, any semantic document can be interacted without losing semantic accuracy. For example, “Fig. 5 → iid-based document shown in Fig. 6 → Fig. 7”.

Figure 8 shows the (1) rules mapped from the received flight reservation document (e.g., Fig. 7) and (2) the automatically generated rules used to execute the document. These rules are used as input to the inference engine in the Tabdoc editor of document receiver. In Fig. 9, each term in the rule is replaced by its concept internal identifier (iid) in CoDic,
which ensures that each term has a clear meaning during reasoning.

IV. EXPERIMENTS
A. DATASET DESCRIPTION
To evaluate the performance of the Tabdoc approach, the experiments will use two datasets.

Dataset 1: The data source comes from the data set Troy200 [45]. It builds 10,000 forms based on 10 different government statistics sites. Figure 10 shows some samples from the data set.

Dataset 2: It includes 97 Excel/pdf spreadsheets from 7 different real-world sources. These resources are available at http://cells.icc.ru/.
B. EXPERIMENTS ON EXTRACTION OF SEMANTIC COMPONENTS AND LOGICAL STRUCTURE

The purpose of this experiment was to measure the effectiveness of the Tabdoc method in extracting and identifying semantic components from 1000-10000 document data sets. Table 4 compares the semantic extraction performance between Tabdoc method and TabbyXL [45], [46] on dataset 1 in terms of accuracy and recall rate. The objects of semantic extraction involve: “entry”, “label”, “entry-label pair”, and “label-label pair”. Each document dataset has the ground truth for each of these four semantic objects. Figure 11 shows that the Tabdoc method is superior to the TabbyXL method in extracting semantic information for entry and entry-label pairs. In the semantic information extraction of label and label-label pair, the performance of the two methods is roughly equivalent. This is mainly because the TabbyXL
method focuses on analyzing the style information and relative position relationships between cells in the table. However, the Tabdoc approach not only analyzes this information, but it mainly focuses on the semantic relationships between cells.

Figure 12 and Figure 13 compare the Tabdoc method with Kim’s method [47] in terms of the performance of logical structure extraction on dataset 2. They are evaluated in terms of the number of tables and cells extracted correctly.

Figures 12 and 13 show that the Tabdoc method improves in terms of the number of tables and cells extracted correctly compared to Kim’s work. This is because the Tabdoc method uses semantic information from tabular documents to extract the logical structure, whereas the Kim method relies only on visual information. For example, Kim’s method cannot resolve complex tables with subtables of various shapes. In addition, it cannot analyze a hybrid unit in a cell that has both attributes and values. In contrast, the Tabdoc approach

| Year | Area harvested | All hay | All hay | All hay | Area harvested | All hay | All hay | Production |
|------|---------------|--------|--------|--------|---------------|--------|--------|------------|
| 1993-1997 | 24,873 | 35,036 | 16,899 | 81,315 | 16,668 | 141,059 |
| 1998-2002 | 24,873 | 35,036 | 16,899 | 81,315 | 16,668 | 141,059 |
| 2003-2007 | 24,873 | 35,036 | 16,899 | 81,315 | 16,668 | 141,059 |

TabbyXL is one of state-of-the-art systems for converting any table in the form of spreadsheet to relational records [45], [46].
can identify logical structures through analyzing cells and the semantic relations between them.

C. EXPERIMENT ON SEMANTIC CONSISTENCY MAINTENANCE (SCM)

The purpose of this experiment is to verify the reliability of the Tabdoc method in delivering the semantics of documents accurately and consistently across contexts. First, the experiment measured the accuracy of the understanding of all the words in each tabular document. To calculate the uncertainty of word comprehension, the evaluation metrics are defined according to the theory of information entropy (IE).

Equation (IV.1) gives the calculation method of vocabulary entropy (VE).

$$\text{Entropy}_t[\log_2(m)] = - \sum_{m_i \in M} p(m_i) \log_2 p(m_i) \quad \text{(IV.1)}$$

where $M$ is the set of different meanings of a word $t$ in a document $doc$, and $p(m_i)$ is the probability of the $i^{th}$ concept of $t$ in $doc$. VE is positively correlated with the uncertainty. That is, the higher the VE of a word, the greater the uncertainty of its meaning in a document, and vice versa. If $t$ has only a concept in $doc$, its VE is zero.

To calculate the uncertainty of document understanding, Equation (IV.2) gives the definition of document entropy.
(DE, \(\text{Entropy}_{\text{doc}}\)). It is equal to the average vocabulary entropy (\(\text{Entropy}_{t}\)) times the average number of words (\(\text{Count}_{t}\)), plus the average relational entropy (RE, \(\text{Entropy}_{rt}\)) times the average number of relations (\(\text{Count}_{rt}\)).

\[
\text{Entropy}_{\text{doc}} = \text{Entropy}_{t} \times \text{Count}_{t} + \text{Entropy}_{rt} \times \text{Count}_{rt}
\]  

(IV.2)

Figure 14 shows that in the process of semantic document interaction, the average “VE” of semantic documents interacted with Tabdoc method is lower than the average “VE” of document conversion implemented with TabbyXL method. In addition, the “VE” of TabbyXL method with CoDic dictionary assisted during document transformation is lower than that of TabbyXL without CoDic. It can be concluded that without a collaborative dictionary (e.g., CoDic) to constrain the semantic expression of words, the uncertainty of word meaning would be increased.

Figure 15 shows the comparison between Tabdoc approach and TabbyXL approach in terms of consistent understanding of semantic documents on dataset 2. In general, a semantic relation is a tuple consisting of two terms and a predicate. The result of the previous experiment shows that the “VE” of Tabdoc method is lower than that of TabbyXL method. Therefore, the average entropy of semantic relations of Tabdoc method is less than that of TabbyXL method. Because the name of the relation can also be treated as a word, there is less uncertainty of a relation in the document interaction implemented by the Tabdoc method. Moreover, the more semantic relations are extracted, the clearer the semantic document is understood and the lower the document entropy is. As shown in Fig. 15,
S. Yang, R. Wei: Semantic Interoperability Through a Novel Cross-Context Tabdoc Representation Approach

Figure 15. Comparison of DE on Dataset 2.

Figure 16. Inference time comparison on Dataset 1.

D. EXPERIMENT ON THE EFFICIENCY OF SIA

The purpose of this experiment is to verify the efficiency of SIA algorithm. It compares the reasoning time between the original Rete algorithm and the “IRA” algorithm over approximately 15,000 entries in dataset 1. During the pre-processing phase, the instantiated value of the two hundred documents were bulk deleted. The query rules are then automatically generated by the document template. The original document instance generates the set of facts.

In this experiment, the Rete algorithm implemented by Jess is used as a comparison. Figure 16 shows that the reasoning time of the original Rete algorithm increases exponentially as the number of query items increases. In contrast, the reasoning time of the improved IRA algorithm increases gently as the number of query items increases. The X-axis represents the number of reasoning documents, and the Y-axis represents the time spent on reasoning. When the number of documents from dataset 1 approaches two hundred, the number of items for semantic reasoning is approximately 15,000.

Another experiment showed that an increase in the number of facts also increases reasoning time. Table 5 shows that when the number of facts approaches $10^5$, the reasoning efficiency of the improved IRA algorithm is higher than that of the Rete algorithm. For example, when four queries were performed on $10^5$ facts, the efficiency of the IRA nearly doubled compared to the Rete algorithm. Figure 17 intuitively shows the trend of reasoning time of the two methods as the number of facts increases.

Table 5. Comparison of inference time on Dataset 2.

| Query number | Fact number | IRA    | Jess  |
|--------------|-------------|--------|-------|
| 4 queries    |             |        |       |
| 10           | 1264 ms     | 1626 ms|
| 100          | 1374 ms     | 1739 ms|
| 1000         | 1470 ms     | 1915 ms|
| 10000        | 2155 ms     | 2391 ms|
| 100000       | 3635 ms     | 4463 ms|
| 13 queries   |             |        |       |
| 10           | 1280 ms     | 1536 ms|
| 100          | 1385 ms     | 1599 ms|
| 1000         | 1455 ms     | 1774 ms|
| 10000        | 2143 ms     | 1872 ms|
| 100000       | 3560 ms     | 8428 ms|

V. COMPARISON WITH RELATED WORK

Table 6 compares current studies on document interaction in terms of document syntax, document template and document instance representation. Comparison results show that...
existing methods have three main limitations. First, it’s not easy to automatically embed and extract semantics in (or from) a document. For example, it is difficult to automatically convert semantic document written in natural language to a machine-processable format (such as RuleML [48], [49]). Second, it is time-consuming and heavy-computing to construct and parse semantic documents. For example, [50], [51] proposes a solution of semantic disambiguation by annotating XML documents (e.g., elements and values) with a machine-readable semantic network (e.g., WordNet) as a common knowledge base. Third, it is not easy to maintain semantic consistency between heterogeneous information sources. For example, to achieve interoperability, [32] requires precise mapping between entities in different ontologies. However, it is difficult to achieve a hundred percent semantic match between diverse domain-wide ontologies. Reference [52] requires semantic similarity computation between keywords in a received document and terms in a specific ontology, which may cause semantic loss under different contexts.

Tabdoc approach proposed can exchange semantic documents across heterogeneous contexts and minimize semantic loss; thereby achieving semantic interoperation between
VI. CONCLUSION

To facilitate cross-context semantic interoperability, this paper proposes a novel Tabdoc approach applying a “divide-conquer strategy” to analyze a complex semantic document from three layers. By using a novel de-contextualization strategy on each layer, document complexity and cross-context problems have been resolved. In the implementation, Tabdoc Grammar and Schema are devised as a context-free document representation syntax. This ensures a semantic document can be syntactically interoperable across domains. To minimize semantic loss and avoid semantic ambiguity, a semantic document is viewed as a table on a whole to restrict semantic expression. Each cell in a table is only allowed to be filled with one concept and the relationship between concepts are depicted by diverse semantic relations. To guarantee semantic disambiguity across domains, each common concept is denoted by a unique concept internal identifier (iid) in CoDic dictionary. CoDic is a dictionary collaboratively created by concept designers. Users can easily employ it by a novel Semantic Input Method (SIM) when designing or editing a semantic document. For semantic understanding, a semantic document is automatically parsed by a novel semantic interpretation algorithm (SIA) that relies on an improved Rete algorithm (IRA) for semantic inference. In a general, the approach proposed in this paper has the following expectations:

- It models a complex semantic document through a structure of matrix tree that simplifies semantic document representation and provides a theoretical methodology to semantic interoperaton.
- A universal semantic document representation language (Tabdoc schema) is designed under Tabdoc grammar, which provides a syntactic foundation for semantic document interoperaton.
- It represents the semantics of a document in different types of rules, which enables a user to create and modify semantics easily and is adaptable to the changing semantic environment in real-world business interaction.

The newly proposed Tabdoc approach provides a scheme where semantic documents can be edited, read, exchanged and processed without rigid standards. We will further investigate the application of Tabdoc approach for information interoperability in the Internet of Things (for example, consistent information interoperation between devices and/or humans) and integration of knowledge graph (for example, semantic integration between different domain knowledge) for smart cities. In addition, it is necessary to extend Tabdoc schema to a more comprehensive language to accommodate more complex semantic document representation.

ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers for their valuable comments and helpful suggestions.

REFERENCES

[1] V. Javidroozi, H. Shah, and G. Feldman, “Urban computing and smart cities: Towards changing city processes by applying enterprise systems integration practices,” IEEE Access, vol. 7, pp. 108023–108034, 2019.
[2] A. Costin and C. Eastman, “Need for interoperability to enable seamless information exchanges in smart and sustainable urban systems,” J. Com- put. Civil Eng., vol. 33, no. 3, May 2019, Art. no. 04019008.
[3] K.-M. Seo, K.-P. Park, and B.-J. Lee, “Achieving data interoperability of communication interfaces for combat system engineering,” IEEE Access, vol. 5, pp. 17938–17951, 2017.
[4] P. Marino, F. Pozza, M. Ubeira, and F. Machado, “Applied information system through the interoperability of data interfaces,” in Proc. 2nd IEEE Int’l Conf. Ind. Informat. (INDIN), 2004, pp. 367–372.
[5] W. Dai, P. Wang, W. Sun, X. Wu, H. Zhang, V. Vyattkin, and G. Yang, “Semantic integration of Plug-and-Play software components for industrial edges based on microservices,” IEEE Access, vol. 7, pp. 125882–125892, 2019.
[6] S. F. Pileggi and A. Voinov, “PERSWADE-CORE: A core ontology for communicating socio-environmental and sustainability science,” IEEE Access, vol. 7, pp. 121717–121788, 2019.
[7] A. Mazayev, J. A. Martins, and N. Correia, “Interoperability in IoT through the semantic modeling of objects,” IEEE Access, vol. 6, pp. 19379–19385, 2018.
[8] G. Bajri, R. Agarwal, P. Singh, N. Georgantas, and V. Issarny, “4W1H in IoT semantics,” IEEE Access, vol. 6, pp. 65488–65506, 2018.
[9] J. Kiljander, A. D’elia, F. Morandi, P. Hyytinne, J. Takalo-Mattila, A. Ylisaakko-Oja, J.-P. Soininen, and T. S. Cinotti, “Semantic interoperability architecture for pervasive computing and Internet of Things,” IEEE Access, vol. 2, pp. 856–873, 2014.
[10] Y. Y. Tang, H. Ma, D. Xi, X. Mao, and C. Y. Suen, “Modified fractal signature (MFS): A new approach to document analysis for automatic knowledge acquisition,” IEEE Trans. Knowl. Data Eng., vol. 9, no. 5, pp. 747–762, Jun. 1997.
[11] E. Y.-T. Lin and C. Zhou, “Modeling and analysis of message passing in distributed manufacturing systems,” IEEE Trans. Syst., Man, Cybern, C, Appl. Rev., vol. 29, no. 2, pp. 250–262, May 1999.
[12] J. Liu, K. He, and D. Ning, “Web service aggregation using semantic inter- operability oriented method,” J. Inf. Sci. Eng., vol. 28, no. 3, pp. 437–452, 2012.
[13] X. Li, Y. Fan, Q. Z. Sheng, Z. Maamar, and H. Zhu, “A Petri net approach to analyzing behavioral compatibility and similarity of Web services,” IEEE Trans. Syst., Man, Cybern. A, Syst. Humans, vol. 41, no. 3, pp. 510–521, May 2011.
[14] J. H. Bremas, M. S. Al-Manir, C. J. Baker, and A. Shaban-Nejad, “A malaria analytics framework to support evolution and interoperability of global health surveillance systems,” IEEE Access, vol. 5, pp. 21605–21619, 2017.
[15] M. S. Benabdelhafid and M. Boufaida, “Toward a better interoperability of enterprise information systems: A cnps and timed cnps-based Web service interoperability verification in a choreography,” Procedia Technol., vol. 16, pp. 269–278, 2014.
[16] V. Apparao, B. Eich, R. Guha, and N. Ranjan, “Action sheets: A mod- ular way of defining behavior for xml and html,” in Proc. World Wide Web Consortium, 1998. [Online]. Available: https://www.w3.org/TR/1998/ NOTE-AS-19980619
[17] G. Xiao, J. Guo, Z. Gong, and R. Li, “Semantic document exchange for E-business: Trends and issues,” in Proc. IEEE 12th Int. Conf. Business Eng., Oct. 2015, pp. 147–153.
[18] P. Qin and J. Guo, “A novel machine natural language mediation for semantic document exchange in smart city,” Future Gen. Comput. Syst., vol. 102, pp. 810–826, Jan. 2020.
[19] G. Xiao, J. Guo, Z. Gong, and R. Li, “Semantic input method of chinese word senses for semantic document exchange in e-business,” J. Ind. Inf. Integr., vol. 3, pp. 31–36, Sep. 2016.
[20] J. Luo, B. Meng, C. Quan, and X. Tu, “Exploiting salient semantic analysis for information retrieval,” Enterprise Inf. Syst., vol. 10, no. 9, pp. 959–969, Nov. 2016.
F. Raml and S. A. Mohd Noah, “Building an event ontology for historical domain to support semantic document retrieval,” Int. J. Adv. Sci. Eng. Inf. Technol., vol. 6, no. 6, pp. 1154–1160, 2016.

J.-U. Heu, I. Qasim, and D.-H. Lee, “FoDoSu: Multi-document summarization exploiting semantic analysis based on social folksonomy,” Inf. Process. Manage., vol. 51, no. 1, pp. 212–225, Jan. 2015.

B. Medjahed, B. Benattallah, A. Bouguettaya, A. H. H. Ng, and A. K. Elmagarmid, “Business-to-business interactions: Issues and enabling technologies,” VLDB J. Int. Very Large Data Bases, vol. 12, no. 1, pp. 59–85, May 2003.

Y. Yan, J. Zhang, and M. Yan, “Ontology modeling for contract: Using OCL to express semantic relations,” in Proc. 10th IEEE Int. Enterprise Distrib. Object Comput. Conf. (EDOC), Oct. 2006, pp. 409–412.

J. Guo, “Inter-enterprise business document exchange,” in Proc. 6th Int. Conf. Electron. Commerce (ICEC), 2006, pp. 427–437.

Y. Charalabidis, F. Lampathaki, and D. Askounis, “Unified data modelling and document standardization using core components technical specification for electronic government applications,” J. Theor. Appl. Electron. Commerce Res., vol. 3, no. 3, pp. 38–51, 2008.

Y. Kabak and A. Dogac, “A survey and analysis of electronic business document standards,” ACM Comput. Surv., vol. 42, no. 3, pp. 1–31, Mar. 2010.

M. Erdmann and R. Studer, “How to structure and access XML documents with ontologies,” Data Knowl. Eng., vol. 36, no. 3, pp. 317–335, Mar. 2001.

J. Shen and Y. Yang, “Extending RDF in distributed knowledge-intensive applications,” Future Gener. Comput. Syst., vol. 20, no. 1, pp. 27–46, Jan. 2004.

J. Guo, “Collaborative conceptualisation: Towards a conceptual foundation of interoperable electronic product catalogue system design,” Enterprise Inf. Syst., vol. 3, no. 1, pp. 59–94, Feb. 2009.

J. Guo, L. Xu, Z. Gong, C.-P. Che, and S. S. Chaudhry, “Semantic inference on heterogeneous E-Marketplace activities,” IEEE Trans. Syst., Man, Cybern. A, Syst. Humans, vol. 42, no. 2, pp. 316–330, Mar. 2012.

M. Rico, M. L. Taverna, M. L. Caliusco, O. Chiotti, and M. R. Galli, “Adding semantics to electronic business documents exchanged in collaborative commerce relations,” J. Theor. Appl. Electron. Commerce Res., vol. 4, no. 1, pp. 72–90, 2009.

R. D. S. Machado, R. B. Almeida, D. Y. L. da Rosa, J. L. B. Lopes, A. M. Ferras, and A. C. Yamin, “EXEHD3A-HM: A compositional approach to explore contextual information on hybrid models,” Future Gener. Comput. Syst., vol. 73, pp. 1–12, Aug. 2017.

K. Taha and R. Elmasri, “XCDSearch: An XML context-driven search engine,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 12, pp. 1781–1796, Dec. 2010.

J. Guo, Collaborative Concept Exchange. Berlin, Germany: VDM, 2008.

C. Xiao, J. Guo, L. Xu, and Z. Gong, “User interoperability with heterogeneous IoT devices through transformation,” IEEE Trans. Ind. Informat., vol. 10, no. 2, pp. 1486–1496, May 2014.

R. Sarno, W. W. Ayu, A. N. Fajrin, D. Manfaat, M. S. Arif, and I. Baihaqi, “Context sensitive grammar for composing business process model variants,” in Proc. Int. Conf. Sci. Inf. Technol. (ICSI), Oct. 2015, pp. 53–57.

J. Kim, D. Jeong, and D.-K. Baik, “A translation algorithm for effective RDB-to-xlml schema conversion considering referential integrity information,” J. Inf. Sci. Eng., vol. 25, no. 1, 2009.

S. Yang, R. Wei, and A. Shigarov, “Semantic interoperability for electronic business through a novel cross-context semantic document exchange approach,” in Proc. ACM Symp. Document Eng., 2018, pp. 1–10.

F. De Saussure, W. Baskin, and P. Meisel, Course in general linguistics. New York, NY, USA: Columbia Univ. Press, 2011.

C. S. G. Khoo and I.-C. Na, “Semantic relations in information science,” Annu. Rev. Inf. Sci. Technol., vol. 40, no. 1, pp. 157–228, 2006.

S. Yang, J. Guo, and R. Wei, “Semantic interoperability with heterogeneous information systems on the Internet through automatic tabular document exchange,” Inf. Syst., vol. 69, pp. 195–217, Sep. 2017.

S. Yang and J. Guo, “A novel approach for cross-context document reasoning in e-commerce,” in Proc. 6th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS), Sep. 2015, pp. 1018–1025.

Z. Ma and X. Wang, “Rule interchange in the semantic Web,” J. Inf. Sci. Eng., vol. 28, no. 2, pp. 393–406, 2012.

A. O. Shigarov, “Table understanding using a rule engine,” Expert Syst. Appl., vol. 42, no. 2, pp. 929–937, Feb. 2015.

A. Shigarov, V. Khristyuk, and A. Mikhailov, “TabbyXL: Software platform for rule-based spreadsheet data extraction and transformation,” Softw.-X, vol. 10, Jul. 2019, Art. no. 100270.

Y.-S. Kim and K.-H. Lee, “Extracting logical structures from HTML tables,” Comput. Standards Interface, vol. 30, no. 5, pp. 296–308, Jul. 2008.

A. Tsadiras and N. Bassiliades, “RuleML representation and simulation of fuzzy cognitive maps,” Expert Syst. Appl., vol. 40, no. 5, pp. 1413–1426, Apr. 2013.

G. Governatori, “Representing business contracts in Rule ML,” Int. J. Cooperat. Inf. Syst., vol. 14, nos. 2–3, pp. 181–216, Jun. 2005.

K. Thrirunaryan, “On embedding machine-processable semantics into documents,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 7, pp. 1014–1018, Jul. 2005.

S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, “The semantic Web: The roles of XML and RDF,” IEEE Internet Comput., vol. 4, no. 5, pp. 63–73, Jun. 2000.

R. Costa and C. Lima, “Knowledge representations with ontology support for collaborative engineering in architecture engineering and construction,” J. Inf. Technol. Construct., vol. 19, no. 26, pp. 434–461, 2014.

L. Carr, T. Miles-Board, A. Woukeu, G. Wills, and W. Hall, “The case for explicit knowledge in documents,” in Proc. ACM Symp. Document Eng., 2004, pp. 90–98.

H. Eriksson, “The semantic-document approach to combining documents and ontologies,” Int. J. Hum.-Comp. Stud., vol. 65, no. 7, pp. 624–639, Jul. 2007.

L. Bing, S. Jiang, W. Lam, Y. Zhang, and S. Jameel, “Adaptive concept resolution for document representation and its applications in text mining,” Knowl.-Based Syst., vol. 74, pp. 1–13, Jan. 2015.

V. Rodríguez-Doncel, J. Delgado, S. Llorente, E. Rodríguez, and L. Boch, “Overview of the MPEG-21 media contract ontology,” Semantic Web, vol. 7, no. 3, pp. 311–332, Mar. 2016.

G. Boella, L. D. Caro, L. Humphreys, L. Robaldo, P. Rossi, and L. van der Torre, “Eumos, a legal document and knowledge management system for the Web to provide relevant, reliable and up-to-date information on the law,” Artif. Intell. Law, vol. 24, no. 3, pp. 245–283, Sep. 2016.

C. Sáez, A. Bresó, J. Vicente, M. Robles, and J. M. García-Gómez, “An HL7-CDA wrapper for facilitating semantic interoperability to rule-based clinical decision support systems,” Comput. Methods Programs Biomed., vol. 109, no. 3, pp. 239–249, Mar. 2013.

J. Tekli, N. Charbel, and R. Chibeir, “Building semantic trees from XML documents,” J. Web Semantics, vol. 37–38, pp. 1–24, Mar. 2016.

J. Tekli, “An overview on XML semantic disambiguation from unstructured text to semi-structured data: Background, applications, and ongoing challenges,” IEEE Trans. Knowl. Data Eng., vol. 28, no. 6, pp. 1383–1407, Jun. 2016.

H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo, “Mining semantic structures from unstructured free text documents,” in Proc. IEEE Int. Conf. Semantic Comput., Jun. 2014, pp. 84–91.

M.-C. De Marneffe, B. MacCartney, and C. D. Manning, “Generating typed dependency parses from phrase structure parses,” in Proc. 5th Int. Conf. Lang. Resour. Eval. (LREC), Genoa, Italy, vol. 6, 2006, pp. 449–454.