A Novel Approach for Business Document Representation and Processing without Semantic Ambiguity in E-Commerce

Shuo Yang and Jingzhi Guo
Faculty of Science and Technology
University of Macau
Macau, China
{yb37416, jzguo}@umac.mo

Abstract—**Exact document interpretation is very important to semantic document exchange. An essential issue for document interpretation is to maintain syntactic and semantic consistency of the exchanged documents between any two autonomous business communities, where the document sender and receiver have no misunderstanding in using the exchanged documents. Existing approaches to resolving this issue mainly adopt document standards. While these approaches are effective in certain degree, the issue of limited flexibility and evolvement in using standards must be explored and resolved. This paper proposes a multi-viewed tabular document (Tabdoc) approach consisting of a Tabdoc model and a document processing procedure to achieving consistent document interpretation between document writers and readers. In this approach, any document is a table, which is seen as a tree to represent document structure. Concepts, layout and grammatical relations of a same document are separated from the document structure and form different views in Tabdoc model.**

Keywords-semantic document; semantic consistency; multi-viewed tabular document

I. INTRODUCTION

Currently, a technical challenge of document exchange [1] is the lack of consistent document interpretation for document interoperation across domains of involved enterprise information systems (EISs). This is because sending parties and receiving parties are often situated in different semantic communities. Specifically, not all firms participating in an e-marketplace adopt the same vocabulary and meaning before exchanging a message. They provide different syntax to exchange information, and they do not associate the information with consistent semantics which facilitates the mapping between different specifications. Thus, the receiving parties (both computers and human) cannot correctly interpret the structure and meaning carried by the received documents. This hinders the future development of e-business document exchange and much affects the automatic document processing [2], leading to e-business automation unavailable.

Here we use an example to see how semantic heterogeneity affects automatic document processing: Seller A sends a valid offer of fridge to Buyer B and Buyer B confirms the offer by sending back an offer acceptance. In this legally valid offer-acceptance business cycle, the Seller's offer is made in Table I and the Buyer's offer acceptance is made in Table II. Both tables are generated based on their local databases and the messages in exchange. If the two firms have ever cooperated, they may know each other very well. However, if they have never co-operated before, they may have misunderstanding in semantics. For instance, the problem happens when Seller A deems that it sells a mini household refrigerator in US$200, but Buyer B believes it confirms an offer of camping fridge only worth of HK$200. Definitely, this is a legally-flawed offer-acceptance cycle and will cause legal consequences. In an extreme condition such that Buyer B do not understand English, it cannot interpret the details of the offer from Seller A, because concepts are only understandable in the Seller A's own context. Besides, relations between concepts or terms are not elaborated for computers. For example, what the relation between the term 'unit price' and the number '200' is.

TABLE I. A VALID OFFER OF SELLER A

Dear sir/madam, This is an offer about fridge from our company. It is the recent product by incorporating many modern design elements that are more suitable for user experience. This kind of fridge is in orange and has the extremely power of low-temperature control. For the favor of customers, the unit price is only $200 if the order quantity is more than 100 pieces (contains). For more details, you are free to contact us at any time with the offer No. S111 and it is valid before 15/12/2015.

TABLE II. A VALID OFFER ACCEPTANCE OF BUYER B

Dear sir/madam, Our company is very pleased to order the recent fridge product from your company as you mentioned in the last offer sheet. It is in orange color with high quality of temperature control. We are planning to order 100 pieces at each price of $200 with the total amount of $20,000. This acceptance confirms the offer No. S111 on 10/12/2015.

Technically speaking, the above problems can be easily avoided if the offer-acceptance cycle is processed by human. Nevertheless, when a trading process is automatically handled by autonomously developed software systems, the business

978-1-4799-8353-7 /15/$31.00 ©2015 IEEE

document sense disambiguation becomes a tough research problem and must be resolved.

The above example shows that the issue of document interoperability is extremely complex and quite context dependent. The interpretation of a piece of document (e.g., a product specification) relies on different contexts. It is impossible for a document writer to image all contexts of document consumers and a document consumer has difficulty inferring correctly the contexts of document writers. By a simple classification, heterogeneous document representations shown in the example have semantic conflicts in semantic encoding (e.g., term conflicts in definition), relations between concepts (e.g., unit price and 200) and context reference systems (e.g., different interpretations of '200'). Besides, heterogeneous document syntax (i.e., structure) constrains automatic document processing. These problems illustrate that, without a proper mechanism, a business document may not be accurately interpreted and automatically processed by receivers.

Even though many existing document standards have tried different approaches, they are not effective to conquer these problems between two unknown parties in the e-marketplace. In addition, most existing methods import the whole document into memory before document processing, which is not applicable to deal with large ones.

This paper aims to eliminate the semantic inconsistency between document writers and readers by proposing a novel multi-viewed tabular document (Tabdoc) model to build semantic documents which have universal document structure representation for automatic document processing. The approach is based on CONEX [5], where business concepts used in different firms can be collaboratively designed by concept designers and easily used by business users without semantic ambiguity.

The rest of this paper is arranged as follows. Section 2 discusses some related work. Section 3 gives an overview of Tabdoc model for document representation. Section 4 proposes a processing procedure for semantic documents built on Tabdoc model. Section 5 describes an experiment on the approach. Finally, a conclusion is provided.

II. Related Work

A. CONEX

CONEX is a collaborative conceptualization approach to maintain semantic consistency between heterogeneous concept sets used in different EISs. CONEX guarantees that all concepts created, communicated, and used in the e-marketplace are accurate and semantically consistent without ambiguity on the CONEX chain of "reified concept $riid_1$ ⇔ local concept $liid_1$ ⇔ mapping concept ($liid_1$, ciid) ⇔ common concept ciid ⇔ mapping concept (ciid, $liid_2$) ⇔ local concept $liid_2$ ⇔ reified concept $riid_2$." It contributes to a trichotomic view of design, implementation, and use of heterogeneous concepts for semantic consistency maintenance. With this view, concept engineers are responsible for collaborative concept design for common concepts and local concepts in a collaborative-concept-editing system [5]; rule makers implement all executable concepts as ruled concepts or control rules for verbs and adjectives in both common and local levels, and concept users automatically reify these concepts and simply use them.

In the recent, our research group extends CONEX with a near synonym graph (NSG) framework based on WordNet [7] [8] for automating the process of multilingual concept disambiguation in order to find multilingual near synonyms, that is, the semantically equivalent and similar concepts in an initial multilingual vocabulary. The main idea is: for all vocabulary entries that need to be collaboratively edited, they are preprocessed by a near synonym finding process, so that collaborative editors can resolve semantic conflicts between vocabulary entries using sets of near synonyms identified in the preprocessing. CONEX and its importance have already been described in the projects of CONEX [5], collaborative document exchange [1], and collaborative process exchange [6] and will not be elaborated in this paper.

B. Business Document Standards

Business document standards can be classified into eight categories, each of which represents a family of related business standards according to their technical features. First, top-down standardization approaches (e.g., UN/EDIFACT [9]) provide general concepts for business document creation. Users can easily find appropriate business elements in the standard. Second, bottom-up standard approaches (e.g., electronic payment standard [10]) focus on maintaining common and important business elements leaving individual requirements for further extension. Third, hybrid standardization approaches (e.g., UBL [11]) integrate the general feature from top-down standards and the extendable feature from bottom-up standards. Forth, early markup language approaches (e.g., OAGIS [12]) utilize XML to define business document standards, where communicated applications need to share business object documents. Fifth, integrated standardization approaches (e.g., ebXML [13]) create consistent business messages and common business processes in order to achieve automated business transactions. Sixth, transitioned standardization approaches (e.g., HL7 [14]) help EDIFACT-based or other standards to be translated to XML. Seventh, implementation neutral standardization approaches (e.g., CCTS [15] [16]) aim to construct general, concept-leveled business document standards without the consideration of concrete syntactic implementation. Eighth, converging approaches (e.g., UNIFI [17]) merge different business standards if they repeatedly define same concepts when dealing with the same problem in business. Each standardization-based business document standard tries to apply one sharable document designing standard to all heterogeneous EISs of involved parties. Technically, for standardization-based approaches, the semantic consistency maintenance between inter-enterprise business documents is limited to the trading partners that have used the same business document standard. However, outside of these trading partners, business document interpretation may not be accurate. In other words, although these standard approaches are effective in certain degree, they cannot guarantee that what any document reader sees is the exact meaning that any document writer wants to expresses.

III. DOCUMENT REPRESENTATION

In the e-marketplace, most documents can be represented in the form of tabular structure which suffices for meaning understanding in most trading cases [18]. In this paper, a document is, essentially represented as a nested table. Such representation is necessary because only a table can minimize the term ambiguity problem by restricting that one table cell for one term or a new sub-table. When building a table, the meaning of each cell is specified and unambiguous. First, table value cells (e.g., empty table cells) can be restricted by table heading cells (e.g., table cells at the first row) in term of semantics and syntax. Second, there exist different grammatical relations between cells. A grammatical relation refers to a functional relationship between constituents in a document. Third, cells in a table are constructed in a certain form that enables human understanding.

To maintain accurate interpretation of documents between interactive parties, it requires not only the consistent document structure but the consistent semantics of document content. This paper proposes multi-viewed tabular document (Tabdoc) model to build documents exchanged among unknown parties. By this model, a document is represented in three aspects, which are syntactic representation, semantic representation and visual representation. Syntactic representation for a tabular document consists of a set of elements which can form a tree structure [4]. This tree can be alternatively structured as a table with columns and rows. Any node of a tree has a unique correspondent position in a table as a table cell, which is identified as a term identifier (tid) in the tree and also identified by a cell identifier of a table. Visual representation concerns with visual styles of elements in a tabular document. Semantic representation consists of concepts and their grammatical relations. Tabdoc model contains several views in the high-level, including structure view for syntactic representation, layout view for visual representation, concept view and association view for semantic representation.

A. Structure View

In this paper, any document is structured as a tree in syntax for data storage. It is syntactically represented based on a vector tree model [3], as follows:

Definition 1 (*Tree-based Document* "τ"). A document is a tree τ, which is represented as a vector tree:

$$\tau = (I_1^1, I_i^2, \dots, I_i^k, \dots, I_i^n) \quad (1)$$

where (1) each node in τ is represented by *I* with another two notations; (2) *level* of a node in τ is $k \in (1, \dots, n)$; (3) *sibling nodes* are represented by $i \in (0, \dots, m)$ at the same level (e.g., *l*); (4) *parent* of a node at level *k* is represented by a vector $(I_1^1, \dots, I_i^{k-1})$; (5) *children* of a node at level *k* is a set of vectors in the form of $(I_1^1, \dots, I_i^{k+1})$ and (6) *root* of τ is a one dimensional vector (I_1^1) with $k = 1$ and $i = 1$.

For example, a transformation from a vector tree to a two-dimensional table can be found in Fig. 1 with three tree nodes identified by *tid* (i.e., a vector).

Alternatively, a document is also structured as a nested table in syntax for presentation, that is, a nested matrix in mathematics defined as follows:

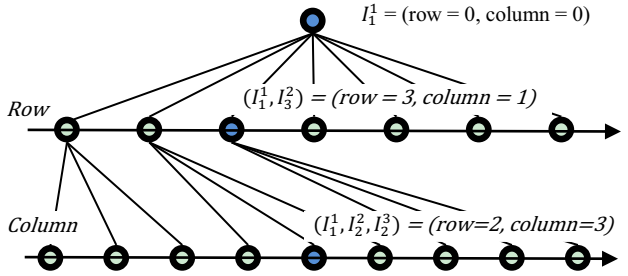

Figure 1. A tree-based document.

Definition 2 (*Table-based Document "t"*). A document is represented as a nested table *t* of below:

$$t = \begin{bmatrix} c_{r_1c_1} & \dots & c_{r_1c_m} \\ \dots & \dots & \dots \\ \dots & c_{r_ic_j} & \dots \\ \dots & \dots & \dots \\ c_{r_nc_1} & \dots & c_{r_nc_m} \end{bmatrix}, c_{r_ic_j=} \begin{bmatrix} c_{r_1c_1} & \dots & c_{r_1c_m} \\ \dots & \dots & \dots \\ \dots & c_{r_ic_j} & \dots \\ \dots & \dots & \dots \\ c_{r_nc_1} & \dots & c_{r_nc_m} \end{bmatrix} \quad (2)$$

where any $c_{r_ic_j}$ is a table cell, which is in the *i* row and *j* column such that $0<i\leqslant$n and $0<j\leqslant$m.

Nested matrixes also enables to construct a table as a tree structure which is called a matrix tree, denoted by $M(n_r, m_c)$. Each element $M(i, j)$ in $M(n_r, m_c)$ refers to the position of a visual cell at *i* row and *j* column (i.e., $c_{r_ic_j}$) in t, which is called matrix position. Matrix position of a cell in t is in the form of: $M[i, j] = M_1[i_1, j_1].M_2[i_2, j_2]\dots M_n[i_n, j_n]$, where $M_n[i_n, j_n]$ is the cell's position in the final matrix that corresponds to the most inside nested sub-table while $M_1[i_1, j_1], \dots, M_{n-1}[i_{n-1}, j_{n-1}]$ correspond to the matrixes that refer to large tables which contain the most inside nested sub-table. Thus, a table-based document can be accessed by traversing its matrix tree.

Vector tree structure (short for tree structure) is responsible for the logical structure of documents, while matrix structure (or table structure) aims for document presentation. Constrains for presentation should not affect the logical structure of a document, but changes of a logical structure often influence the presentation. Thus, we model them separately, but make an internal connection between them as follows.

Syntactically, any tree-based document can be exactly mapped onto a table-based document. Given a tree-based document (τ) and a table-based document (t), τ and t is exactly mapped if and only if:

$$tid(I_i^k) = \begin{cases} table\ root,\ if\ k = 1 \\ (1, i),\ if\ k = 2 \\ (x, i),\ if\ k = 3\ and\ I_i^{k-1} = I_x^{k-1} \\ (x, y).(1, i), if\ k > 3, k\%2 = 0\ and\ M[I_i^{k-1}] = (x, y) \\ (x, y).(z, i), if\ k > 3, k\%2 = 1, M[I_i^{k-2}] = (x, y)\ and \\ M[I_i^{k-1}] = (x, y).(1, z) \end{cases} \quad (3)$$

From (3), it is known that for any I_i^k, if $k = 1$, then I_i^k is a table root. If $k = 2$, it means the parent node of I_i^k imports a list of nodes which correspond to a set of table cells. At this condition, I_i^k maps to the matrix position $M[i, j] = (1, i)$ of a table. If $k = 3$ and the sibling number of the parent node of I_i^k is *x*, it means the grandparent of I_i^k imports a table. At this

condition, I_i^k maps to the matrix position *M[i, j] = (x, i)*. If *k > 3* and *k % 2 = 0*, it means the parent node (I_i^{k-1}) of I_i^k imports a list of nodes. From table view, the table cell corresponding to I_i^{k-1} contains a set of sub-cells. If the matrix position of I_i^{k-1} is *(x, y)*, then the matrix position of I_i^k is *(x, y).(1, i)*. If *k > 3* and *k % 2 = 1*, it means the grandparent node (I_i^{k-2}) of I_i^k imports a table. If the matrix position of I_i^{k-2} is *(x, y)* and the sibling number of I_i^{k-1} is *z*, then the matrix position of I_i^k is *(x, y).(z, i)*.

B. Layout View

Layout view is a style sheet used for describing the looking of a tabular document. It is designed primarily to enable the separation of tabular document content from its presentation by storing presentation instructions in a separate style file. In the following of this paper, matrix position will be used to label the position of a cell in a table-based document (*t*) and a pseudo-table will be built by normalizing each cell of corresponding matrix tree of t. A cell ($c_{r_ic_j}$) situated at the *i* row and *j* column (*M(i, j)*) in t is called a normalized cell if and only if:

$$c_{r_ic_j} = \begin{cases} 1, when\ it\ has\ value \\ 0, when\ it\ is\ empty \\ -1, when\ it\ has\ subtable \\ 1\{x\}^n\ or\ 0\{x\}^n, when\ it\ is\ merged\ with\ n\ right\ cells \\ \{x\}^n 1\ or\ \{x\}^n 0, when\ it\ is\ merged\ with\ n\ below\ cells \end{cases} \quad (4)$$

Following the theory of [19], it is important to separate logical structure (tree structure) and presentational structure (table structure). First, logical structure is a prerequisite to automate document processing, which can be manipulated independently of presentational structure. For example, to add or remove a node in a tree structure, we no longer have to determine which cells should be added or removed from the presentation. Second, by associating different layout and styles with a tree structure, a table can have various presentations. Besides, tree structure cannot well describe presentational feature of a tabular document, since a same tree structure may correspond to different tabular formats. Therefore, this paper imports the concepts of matrix tree and normalized cell to assist tabular presentation.

As nodes in a tree structure of a document can be added or deleted if needed, the value of each normalized cell in any matrix of a matrix tree can be updated at any time. This means tree structure and matrix tree, both corresponding to a same document, will be updated together if the document is modified. For example, a table (t') is given below where we simplify *tid* by only considering sibling number as shown in Table III.

TABLE III. AN EXAMPLE OF A TABLE

<table>
<tr><td>0.k.1.0 ($c_{r_1c_1} = 1$)</td><td colspan="3">0.k.1.1 ($c_{r_1c_2} = 1$)</td></tr>
<tr><td>0.k.2.0 ($c_{r_2c_1} = 1$)</td><td colspan="3"><table><tr><td>1.0</td><td>1.1</td><td>1.2</td></tr><tr><td>2.0</td><td>2.1</td><td>2.2</td></tr><tr><td>3.0</td><td>3.1</td><td>3.2</td></tr></table></td></tr>
<tr><td>0.k.3.0 ($c_{r_3c_1} = 1$)</td><td colspan="3">0.k.3.1 ($c_{r_3c_2} = 1$)</td></tr>
<tr><td>0.k.4.0 ($c_{r_4c_1} = 1$)</td><td colspan="3">0.k.4.1 ($c_{r_4c_2} = 1$)</td></tr>
</table>

The pseudo-table of t' is $t' = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$, where $c_{r_2c_2} = -1$ which means that this cell imports another sub-table such that $c_{r_2c_2} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

Layout view has a simple syntax and uses English keyword 'Style' as the beginning to specify various style properties followed by the position of a cell ($cid(x,y)_{M(x,y)}$) in a tabular document (see (5)).

$$Style(cid(x,y)_{M(x,y)}) = \{property_1 = value_1, \ldots, property_n = value_n\} \quad (5)$$

C. Concept View

Concept view consists of a set of concepts used for reifying cells in a tabular document. It plays a role of database to store and display content in a table. A tabular document consists of a set of reification relations. Each reification relation consists of a $cpt(x, y)_{M(x,y)}$ that declares which cell a reification relation applies to by matching a cell's position in a tabular document and a declaration block. Formula (6) shows the syntax of concept view.

$$Concept[cpt(x, y)_{M(x,y)}] = \{\ concept \mid term\ \} \quad (6)$$

D. Association View

Association view is a set of grammatical relations used for associating concepts in a tabular document. Typically, a grammatical relation is of a particular type that specifies in what sense an object is related to another objects in a document. In a tabular document, several cells can form different kinds of relation in terms of different grammatical relation types. In this paper, ten types of grammatical relations are considered. Specifically, *subclass relation* defines which objects are classified by which class. *Part-of relation* defines which objects can be combined together to form a composite object. *Causality relation* defines which object is the cause of which object (effect). *Reference relation* defines which objects are the further explanations of which object. *Calculation relation* defines which objects can form a mathematical or logical operation together. *Parallel relation* defines which objects have the same superiority. *Progressive relation* defines the superiority of which objects is progressively increase (default) or decrease. *Sequential relation* defines which objects have an order among them. *Instance relation* defines which object (instance) is the reification of which object (reified). *User-defined relation* defines domain-specific relations. Formula (7) shows the syntax of the association view.

$$Association[ass(x,y)_{M(x,y)}] = \{<grammatical\ relation\ type>, ass_1(x_1,y_1)_{M(x1,y1)}, \ldots, ass_n(x_n,y_n)_{M(xn,yn)}\} \quad (7)$$

E. Mapping between Different Views

Structure view gives the information about positional and nested features for each element in a document. It uses vector tree to locate nodes in the tree structure of the document and utilizes matrix position to identify cells in the tabular structure of the same document. When any term identifier (*tid*) in a

vector tree is one-to-one correspondence with a matrix position in a matrix tree, then they describe the same document.

From the syntax of layout, concept and association view, all these views use matrix position to locate elements in a pseudo-table before specifying how a particular feature can be applied to a specific cell in a tabular document. In a matrix tree, each element $M(r_i, c_j)$ refers to the position of a visual cell at *i* row and *j* column in a tabular document. Thus, if the matrix position (e.g., *M(i, j)*) of a cell in the syntax of a view (layout, association or concept view) is equal to a tid, then the feature specified by the view will be applied to the particular node in the vector tree of the structure view. Fig. 2 shows the mapping model between different views.

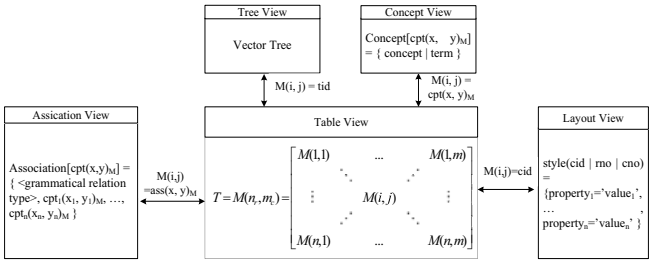

Figure 2. Mapping model.

IV. DOCUMENT PROCESSING

The procedure of semantic document processing in Tabdoc approach consists of six steps such that syntactic check, semantic check, document analysis, document understanding, document template selection and new document creation. In the following, these steps are discussed in detail.

A. Syntactic Check

Syntactic check is to examine whether a document is consistent with Tabdoc schema. Tabdoc schema is a messaging standard for creating documents which are interoperable in the syntax at both document writers' and readers' sides. It defines a semantic document as a set of recursive abstract concepts and/or reified concepts as follows.

Document:=$concept_1[(denoter_1,...denoter_n)\{value\}](concept_2[...],...,concept_m[...])$

In this schema, any abstract concept comes from the CONEX [5] and can form a hierarchical structure by nesting other abstract concepts. It represents a semantic object if and only if it is reified as any {value}. It denotes itself by using a denotation structure […] with a set of denoters.

If syntactic check is not proven, it will block the unproven part with a consequence of either aborting or continuing the process, depending on predefined procedural rules at the document receiver side. Document receivers can design syntactic check in the form of if-then structure like:

$$If \forall s'_{(@tid=i)} \in D' \; ! = Tabdoc\ schema \in D\ Then\ Reject(D') \quad (8)$$

The notation "!=" means not satisfy. Formula (8) means if any element (s') at *tid=i* in a received document (D') not satisfies user-defined Tabdoc schema of the original document (D) from a document writer, then the document receiver will reject D'.

B. Semantic Check

When arrived, the concepts and grammatical relations in the received document will be translated into the semantically equivalent concepts and relations in the context of the receiver according to the CONEX chain [5]. Semantic check aims to examine whether the incoming document is consistent with mutually understandable CONEX concepts by validating whether the vocabularies used in the document are semantically consistent with common vocabularies in CONEX. If exists vocabularies in the received document not pre-defined in CONEX, it means that the document sender used some its own vocabularies to construct the document, which may lead to semantic inconsistency at the receiver side. In this case, the semantic check will block the unproven part with a consequence of aborting the process and a request for constructing new vocabularies in CONEX from the receiver will be sent to the document writer.

Since synonyms are considered as semantically consistent concepts across heterogeneous EISs, they are semantically equivalent in CONEX if they have the same iid [5]. In CONEX, different meanings of a polysemous concept have different iids, therefore they are not semantically equivalent even if they use the same word form. Thus, synonyms do not affect semantic check but polysemous concepts do. Document receivers can also design semantic check in the form of if-then structure like:

$$If \forall v'_{(@tid=i)} \in D' \; !=_{sem} v \in CONEX\ Then\ Request(v') \quad (9)$$

The notation "$!=_{sem}$" means not semantically equivalent. Formula (9) means if any vocabulary (*v'*) at *tid=i* in D' not semantically equivalent to a vocabulary (*v*) in CONEX, then the document receiver will request the sender to construct a new vocabulary in CONEX for accurate interpretation.

C. Document Analysis

Document analysis is to extract the document structure. Since Tabdoc model follows sign theory [5] by structuring a document as a tree structure, document analysis results in creating a vector tree of the incoming document. Given a document whose root is identified by 1.i.j…x.y.z where *tid* is simplified by only considering sibling number for easy reading, a vector tree can be reconstructed through the following processes:

Step 1: Set the document root as a vector tree root = 1.i.j…x

Step 2: Create the row nodes indexed from 1.i.j…x.1 to 1.i.j…x.y based on the table row number (e.g., y).

Step 3: Create the column nodes indexed from 1.i.j…x.1.1 to 1.i.j…x.y.z based on the table column number (e.g., z).

Step 4: Create sub-tree by first identifying a sub-table root. By setting a convention, a table root shall be explicitly identified by the denoter of a sign.

Step 5: Repeat the above steps until all terminal nodes have no sub-tree.

For example, given a table below by supposing the root of the table is 1.i.k, a vector tree (VT) will be immediately constructed as shown in Fig. 3.

1.i.k(0, 0)	Not Exist	Not Exist	Not Exist	Not Exist	Not Exist
1.i.k(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)
1.i.k(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)
1.i.k(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)
1.i.k(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)

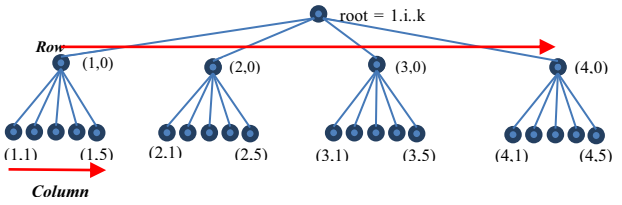

Figure 3. An example of creating a vector tree by a table.

D. Document Understanding

Document understanding deals with finding reification and grammatical relations by using the concept view and association view of the incoming document. For each reification relation in concept view, a concept will instantiate a node in the vector tree through the mapping between vector tree and matrix tree. For each grammatical relation in the association view, a link will be drawn between related nodes in the vector tree. In a tabular document, several cells can form different kinds of relation in terms of different grammatical relation types. For example, Table IV lists parts of the grammatical relations in Fig. 3.

TABLE IV. GRAMMATICAL RELATIONS IN FIG. 3

Association[ass(1,0)$_{M(1,0)}$]={<part-of>, ass(1, 1)$_{M(1,1)}$}	Association[ass(2,0)$_{M(2,0)}$]={<part-of>, ass(1, 1)$_{M(1,1)}$}
Association[ass(3,0)$_{M(3,0)}$]={<part-of>, ass(1, 1)$_{M(1,1)}$}	Association[ass(4,0)$_{M(4,0)}$]={<calculation>, *, ass(2,0)$_{M(2,0)}$, ass(3, 0)$_{M(3,0)}$}

E. Document Template Selection

Document template selection is to choose an appropriate document template according to the type of the incoming document in order to create a new document as a feedback. Thus, all participating EISs in the e-marketplace need to maintain a template library respectively to store kinds of document templates. Document template selection is implemented in a hybrid collaborative human-agent framework, which mixes with human and automated agents. In this framework, humans are responsible for providing the human-related work, e.g., template creation, modification and publication. Automated agents are responsible for non-human work, e.g., template matching, choosing and matrix tree creating.

F. New Document Creation

A new document will be created after mapping related nodes of the reified vector tree of the incoming document to proper positions of the matrix tree of the selected document template. The mapping procedure is to index each related node of the vector tree of the exact matrix position in the matrix tree. For example, if the matrix position of the matrix tree of a document template (T) is the transposition of that of the received document, the mapping procedure is as below.

$$T = \begin{bmatrix} M(1,1) & \dots & M(1,m) \\ \dots & \dots & \dots \\ \dots & M(i,j) & \dots \\ \dots & \dots & \dots \\ M(n,1) & \dots & M(n,m) \end{bmatrix}, M(i,j) \in T = tid_{1.j.i} \in VT$$

Next, the new document can be further edited before transferring to appropriate receivers according to specific business processes.

V. EXPERIMENTS AND RESULTS

A. Document Representation Language

Tabdoc model regards any document as a tabular document by using a vector tree and matrix tree as its document structure. Each node in a vector tree is indexed by a vector concept which represents the position of a cell in a tabular document. Tabdoc model follows sign theory [5] by structuring a document as a compound sign. A compound sign consists of a set of compound signs until each compound sign is a list of atomic signs. Specifically, any document which can be represented as a tabular document and its document structure can form a vector tree with each node indexed by a vector concept is called sign-oriented document. At present, sign-oriented documents are implemented by XML. To facilitate sign-oriented documents transformable into tabular documents, some denoters must be predefined as follows:

Sign(
- ***tid***, a sign identifier in a document to specify the position of the sign in a tree-based document. Its format is x.y…z
- ***obj***, an object type. When *obj* = "*table*", a sign introduces a table and this sign becomes table root. When *obj* = "*cell*", a sign introduces a cell. When *obj* = "*list*", a sign introduces a list of cells.
- ***term***, a name of a sign to specify the meaning of the sign.
- ***ref***, a reference of a term to a common vocabulary.
- ***ass***, an association of a sign to specify which signs can form a grammatical relation.
- ***style***, a format specification of a sign in order to polish the looking of the *term* in the sign.

)

The two denoters such that *tid* and *obj* are designed for tree view and table view, where *obj* aims to specify how a vector tree can be mapped onto a table. These two denoters are effectively to create and locate any sign element in a sign-oriented document. Fig. 4 is an example of a sign-oriented document where indexes in the form of 0…i are vector concepts. Every sign element in a sign-oriented document is identified by a term identifier (i.e., vector concept) and the semantics of each sign element is assigned by a set of denoter and value pairs. The whole tree-structured sign-oriented document tells the structure of a document.

```
<sign tid = "0" obj="table"> // tid = 0, root of a vector tree
        <sign tid = "0.1"> // row 1
                <sign tid = "0.1.1"/> // column 1
        </sign>
        <sign tid = "0.2"> // row 2
                <sign tid = "0.2.1"/> // column 1
        </sign>
        <sign tid = "0.3"> // row 3
                <sign tid = "0.3.1"/> // column 1
        </sign>
        <sign tid = "0.4"> // row 4
                <sign tid = "0.4.1"/> // column 1
        </sign>
```

```
</sign>
```

Figure 4. An example of a sign-oriented document.

Given <sign tid = "0" obj = "table"/>, it is immediately known that a tree structure will be drawn from the position tid = 0 as shown in Fig. 5 and its matrix tree is $\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$.

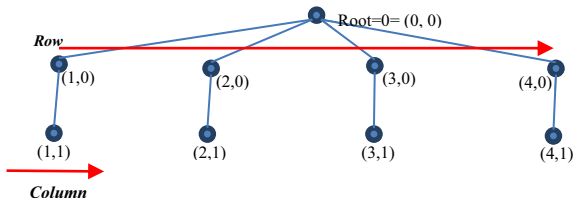

Figure 5. Vector tree of the sign-oriented document in Fig. 4.

B. An Example based on Tabdoc Model

In this paper, a system called Tabdoc Editor is developed to implement Tabdoc approach. This section shows an example about how to represent a purchase order based on the Tabdoc model. Fig. 6 shows the prototype of Tabdoc approach. In Fig. 6, ConexNet is responsible for providing terms (i.e., collaborative signs, call cosigns) in a CONEX dictionary to all document writers and document readers for them to use through a semantic input method (SIM) which is an input method engine. CONEX dictionary is kept updated in real-time. A document writer writes (or edits) a semantic document through a Tabdoc Editor to form a tabular document, which is again transformed into a sign-oriented document or a compound sign that is a set of hierarchically arranged atomic signs. The compound sign is then sent to the remote document system in other contexts.

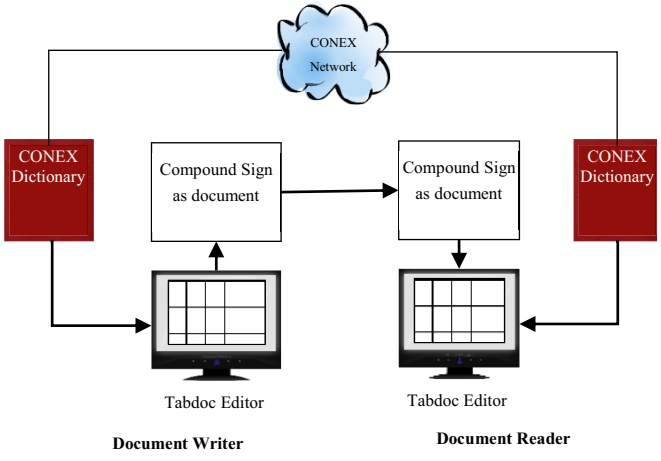

Figure 6. The prototype of Tabdoc approach.

In the process of document construction, there exist two kinds of roles such that document template designers and document writers. One similarity between them is they both use Tabdoc Editor as the developing environment. However, the former are mainly responsible to construct document structure (i.e., tree structure and table structure), render the format (i.e., layout view) and fill in abstract concepts and desirable relations (i.e., concept view and association view) in order to complete document templates. The latter are responsible for reifying existed document templates (e.g., Fig. 7) with required values. Another is they both use CONEX as a term provider through SIM. Tabdoc Editor will automatically summarize different views of a tabular document when it has been created, as shown in Fig. 8.

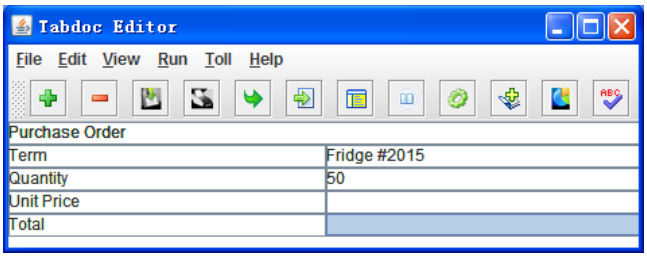

Figure 7. An example of a purchase order.

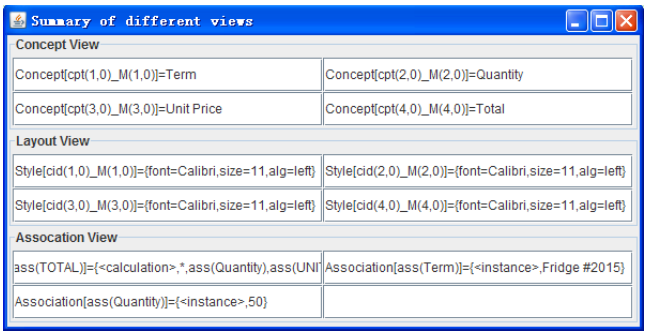

Figure 8. Summary of different views.

After a Tabdoc document is arrived at a document receiver, document processing is executed to test whether the document sent by the document writer faithfully arrives at the document reader's side and whether the semantics of the document is consistent during transferring. In the above example, after a semantic document template is reified by a document writer, it is transformed into a sign-oriented document that is a set of hierarchically arranged atomic signs. The compound sign is then sent to the Tabdoc Editor of the document receiver. When arrived, the concepts and grammatical relations in the received document will be translated into the semantically equivalent concepts and relations in the context of the receiver according to the CONEX chain [5]. Then the Tabdoc Editor of the receiver will first check the syntactic and semantic consistency of the document according to predefined rules like (8) and (9). After that, document structure and different views of the document will be extracted as shown in Fig. 5 and Fig. 8 and the document will be presented on the Tabdoc Editor of the receiver as shown in Fig. 7. Next, the automated agent of Tabdoc Editor will choose an appropriate template ready to respond to the purchase order as shown in Fig. 9. After mapping necessary node of the vector tree of the incoming document to the proper position of the matrix position of the template and completing required authoring, the reified document template (see Fig. 10) will be sent out according specific business processes.

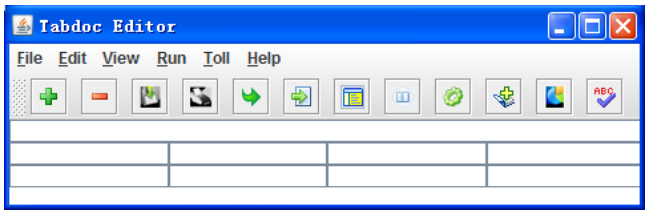

Figure 9. An example of template.

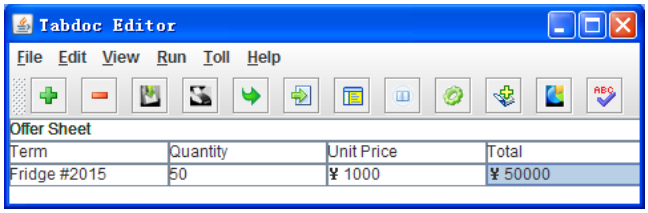

Figure 10. Reified template as feedback document.

Experiment results show that document writers can autonomously design their own tabular document templates without affecting readers' reading and terms used by the document writer are semantically equivalent to the terms presented to the document reader.

VI. CONCLUSION

Currently, it is difficult to interpret a same semantic document in different EISs to conclude with a same meaning and automatically make a feedback. Even though existing document standards have tried many approaches, they are not effective to conquer this problem. The receiving parties (both computers and human) may not correctly interpret the meaning carried by the received documents because sending parties and receiving parties are often in different semantic communities. To achieve exact meaning interpretation between document writers and readers, this paper proposes Tabdoc approach with a processing procedure to respectively represent and process documents while maintaining syntactic and semantic consistency. The novelties of our method are: (1) the autonomy of document writers is enabled, such that document writers can autonomously design any document templates without affecting document readers to read and use the information; (2) memory is saved. This benefit comes from the dynamic generation of nodes in a vector tree. Blank cells of a table will have no nodes in a vector tree, which means there will have no continuous *tid* for sibling nodes; (3) semantics is added into documents by using the common vocabulary CONEX and grammatical relations, and the meaning interpretation between document writers and readers are kept consistent; (4) document analysis is fully automatic and schema-independent, relying on no background information about the vocabulary and the actual textual content of the document; (5) a new document can be automatically created as a feedback at the end of document processing.

ACKNOWLEDGMENT

This research is partially supported by University of Macau research grants no. MYRG069(Y1-L2)-FST12-GJZ and no. MYRG2015-00043-FST.

REFERENCES

[1] J. Guo, "Inter-Enterprise Business Document Exchange," In *Proceedings of the 8th International Conference on Electronic Commerce*, ICEC'06, ACM Press, pp. 427-437.

[2] Angelo D. Iorio, S. Peroni, F. Poggi, D. Shotton, and F. Vitali, "Recognising document components in XML-based academic articles," In *Proceedings of the 13th ACM symposium on Document engineering* (Sept. 2013), pp. 181-184.

[3] J. Guo, "SDF: A Sign Description Framework for Cross-context Information Resource Representation and Interchange," In *Proceedings of the 2nd Int'l Conf. on Enterprise Systems* (Aug. 02-03, 2014), pp. 255-260.

[4] J. V. Dury, "Using RDFS/OWL to ease Semantic Integration of Structured Documents," In *Proceedings of the 13th ACM symposium on Document engineering* (Sept. 10-13, 2013), pp. 189-192.

[5] J. Guo, "Collaborative conceptualization: Towards a conceptual foundation of interoperable electronic product catalogue system design," *Enterprise Inf. Syst.* 3, 1 (Feb. 2009), pp. 59–94.

[6] J. Guo, Z. Hu, C.-K. Chan, Y. Luo, and C. Chan, "Document-oriented heterogeneous business process integration through collaborative e-marketplace," In *Proc. of Tenth International Conference on Electronic Commerce* (Innsbruck, Austria, August 19-22). ICEC'08. ACM Press.

[7] G. Miller, "WordNet: A lexical database for English," *Commun. ACM.* 38, 11 (Nov. 1995), pp. 39–41.

[8] WordNet. Available: http://wordnet.princeton.edu/

[9] Ma Yongheng, Xiong Qianxing, Wu Yefu, Meng Bo, Tian Jie, and Yang Li'na, "Build W3C XML Schema for UN/EDIFACT Messages with Multilayer and Modular Architecture," *International Conference on Web Services* (June 23 - 26, 2003), pp. 515-519.

[10] Antonio Ruiz-Martínez, Óscar Cánovas Reverte, and Antonio F. Gómez-Skarmeta, "Payment frameworks for the purchase of electronic products and services," *Computer Standards & Interfaces*. 34, 1 (Jan. 2012), pp. 80-92.

[11] UBL. Universal Business Language. http://www.oasis-open.org/committees/ubl/.

[12] OAGIS. Open Application Group Integration Specification. http://www.oagi.org.

[13] Alessio Bechini, Andrea Tomasi, and Jacopo Viotto, "Document Management for Collaborative e-Business: Integrating EBXML Environment and Legacy DMS," In *Proceedings of the International Conference on e-Business* (July 28-31, 2007), pp. 78-83.

[14] Fabrizio Pecoraro, Daniela Luzi, and Fabrizio L. Ricci, "A conceptual Framework to Design a Dimensional Model Based on the HL7 Clinical Document Architecture," *Studies in Health Technology and Informatics*. 205 (2014), pp. 278-282.

[15] Danijel Novakovic and Christian Huemer, "Applying business context to calculate subsets of business document standards," *Information Technology and Management* (April 24, 2015), DOI= http://dx.doi.org/10.1007/s10799-015-0228-2.

[16] Konstantin Hyppönen, Miika Alonen, Sami Korhonen, and Virpi Hotti, "XHTML with RDFa as a Semantic Document Format for CCTS Modelled Documents and Its Application for Social Services," *ESWC 2011Workshops*. 7117 (2012), pp. 229-240.

[17] UNIFI. Universal financial industry message scheme. http://www.iso.org/iso/catalogue_detail?csnumber=55005.

[18] Jingzhi Guo and Guangyi Xiao, "Achieving Meaning Understanding in E-Marketplace through Document Sense Disambiguation," In: Software Services for e-World - I3E, IFIP AICT 341 (2010), pp. 127-138.

[19] Horst Silberhorn, "TabulaMagica – An Integrated Approach to Manage Complex Tables," In *Proceedings of the 1th ACM symposium on Document engineering* (Nov. 9-10, 2001), pp. 68-75.