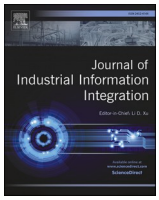
Journal of Industrial Information Integration 38 (2024) 100547

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.sciencedirect.com/journal/journal-of-industrial-information-integration



Full Length Article

Employing word-embedding for schema matching in standard lifecycle management

Hakju Oh [a,*], Boonserm (Serm) Kulvatunyou [a], Albert Jones [a], Tim Finin [b]

[a] Systems Integration Division, National Institute of Standards and Technology, USA
[b] Department of Electrical and Computer Engineering, University of Maryland Baltimore County, USA

ARTICLE INFO

Keywords:
Semantics
Integration
Standards
Word-embedding
Semantic-matching
Frameworks

ABSTRACT

Today, businesses rely on numerous information systems to achieve their production goals and improve their global competitiveness. Semantically integrating those systems is essential for businesses to achieve both. To do so, businesses must rely on standards, the most important of which are data exchange standards (DES). DES focus on technical and business semantics that are needed to deliver quality and timely products and services. Consequently, the ability for businesses to quickly use and adapt DES to their innovations and processes is crucial.

Traditionally, information standards are managed and used 1) in a platform-specific form and 2) usually with standalone and file-based applications. These traditional approaches no longer meet today's business and information agility needs. For example, businesses now must deal with companies and suppliers that use heterogeneous syntaxes for their information. Syntaxes that are optimized for individual but have different objectives. Moreover, file-based standards and the usage specifications derived from the standards cause inconsistencies since there is neither a single standard format for each usage specification nor a single source of truth for all of them.

As the number and types of information systems grow, developing, maintaining, reviewing, and approving standards and their derived usage specifications are becoming more difficult and time consuming. Each file-based usage specification is typically based on a different syntax than the standard syntax. As a result, each usage specification must be manually updated as the standard evolves; this can cause significant delays and costs in adopting the new and better standard versions. National Institute of Standards and Technology (NIST) in collaboration with the Open Application Groups Inc. (OAGi) has developed a web-based standard lifecycle management tool called SCORE to address these problems. The objective of this paper is to introduce the SCORE tool and discuss its particular functionality where a word-embedding technique has been employed along with other schema-matching approaches. Together they can assist standard users in updating the usage specification due to the release of new version of a standard leading to faster adaptations of DES to new processes.

1. Introduction

Mass customization is becoming commonplace in many, modern, manufacturing industries [1]. To be competitive, those industries must develop, in a timely and cost-effective fashion, new products, processes, and services. All of which must be based on a variety of emerging, digital technologies. Furthermore, these technologies have become the foundation for two important, manufacturing strategies: Industrie 4.0 and Smart Manufacturing [1–4]. Implementing these strategies, however, involves the integration of both the physical and the digital components. Examples of those components include automated robots, machining processes, cloud computing, data analysis (DA), and artificial intelligence (AI). The digital components, which can be viewed as information models and systems, interoperability has been recognized as an essential feature for integration. Moreover, the predominant approach to facilitating interoperability is adopting well-designed, canonical, data models written in interchangeable formats, called *data exchange standards (DES)* [5]. Standards-based, data exchange, as opposed to the usual, point-to-point exchange, allows scalable integration among supply-chain-trading partners, software systems, process controllers,

* Corresponding author at: Systems Integration Division, Engineering Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.
E-mail address: hakju.oh@nist.gov (H. Oh).

https://doi.org/10.1016/j.jii.2023.100547

Available online 29 December 2023

machines, and devices [5–8]. DES are typically developed and maintained through standards development organizations (SDOs) with contributions from supply-chain partners, manufacturers, and software vendors. In this way, DES provides a shared, interoperable, data model that can enable scalable and cost-effective integration.

Traditionally, DES are managed and used in a platform-specific form and, usually, with standalone or file-based applications. Unfortunately, this traditional approach does not meet today's agility needs. Those needs exist because businesses now must deal with fast changing business requirements and heterogeneous syntaxes. Each of which is optimized for individual and, possibly, different, production objectives. Moreover, file-based standards, and their usage specifications, were derived from the earlier standards that often cause inconsistencies. Those inconsistencies arose because there was neither a standard format for each usage specification nor a single source of truth for all of them. In addition, each file-based usage specification is typically based on a different syntax than the standard syntax. As a result, each usage specification must be manually updated as the standard evolves, causing significant delays and costs in adopting the new and better versions of the standard. Finally, as the number and types of information systems continue to grow developing, reviewing, and approving new standards and creating and maintaining new usage specifications are becoming more difficult.

To address these problems, NIST has been collaborating with the OAGi, a DES development organization, to create a web-based, standard, lifecycle-management tool called SCORE. This paper introduces the tool and discusses a particular function called Business Information Entity (BIE) [9] Uplifting in the tool. The BIE Uplifting function is a function that assists a user in updating the usage specification – whenever such an update is required. It is a utility built-on top of digitalization and communalization of DES and its usage specification. Although together they address many of the issues outlined earlier, migrating a usage specification from one standard release to another can still be a difficult task. That is because the migration requires schema matching. In this research, we explore and outline how a relatively powerful natural language process technique, namely word-embedding, may be engineered to assist users in a user-interactive, web-based application where response time and limited computing resource is of essence.

The rest of the paper is structured as follows. Section 2 provides a brief overview of the SCORE tool with the main objective of introducing the BIE Uplifting function. Section 3 provides a literature review on schema matching. Section 4 describes the engineering and analysis of schema matchers employed in this research. Section 5 illustrates the implementation in SCORE. Finally, Section 6 provides a conclusion and remark.

2. Overview of the score tool

The SCORE tool employs the international standard for platform-independent representations called Core Component Specification (CCS). CCS is both an ISO standard [10] and a UN/CEFACT standard [9]. Although there are slight variations between the two standards, SCORE is compatible with both. SCORE realized the data model, associated with these standards, in a relational database to allow for improved data management.

SCORE consists of two categories of functions 1) one to develop and maintain a DES and 2) one to develop and maintain usage specifications. The Open Applications Group Integration Specification (OAGIS), also known as ConnectSpec [11], a standard for enterprise and supply chain integrations, has been used extensively to validate those two functions. Today, SCORE is a mature tool. It is used officially for developing and maintaining the OAGIS standards; and it is deployed in large enterprises to manage their integration problems using OAGIS. In addition to the OAGIS standard, SCORE is being piloted for the ADAPT (and Agriculture DES) [12,13].

Usage specification, the focus of the second category of functions, is an information profile derived from different parts of the existing DES. As an information-based model, each specification provides a specific integration context. That model usually contains a subset of entire set of data elements; and each data element, in that subset, may have a definition and a value domain. Both of which are more refined than those in the existing DES. When combined, they provide more specific instructions to each code developer, who must implement the standard in their specific, and sometimes different, integration environment. In this case, different environment includes different types of middleware, business applications, and business and engineering processing tools.

Fig. 1 illustrates two usage specifications of a DES object called Inspection Order. The left one is associated with the raw-material, testing context and the right one is associated with the finished-goods, testing context in a large food manufacturing enterprise. Fig. 2 illustrates a contextual definition of the Identifier field (selected in Fig. 1) instructing developers that it should be populated with a specific type of identification content from the Labs Information Management System (LIMS) [14]. LIMS is a sample tracking system typically used in a food and a drug manufacturing.

SCORE can represent usage specifications as Business Information Entity (BIE) according to CCS. In the next section, BIE is described since its data will be used in examples when discussing 'schema matchers'. Then, in the subsequent section, the BIE Uplifting function is introduced.

2.1. BIE (Usage specification) representation

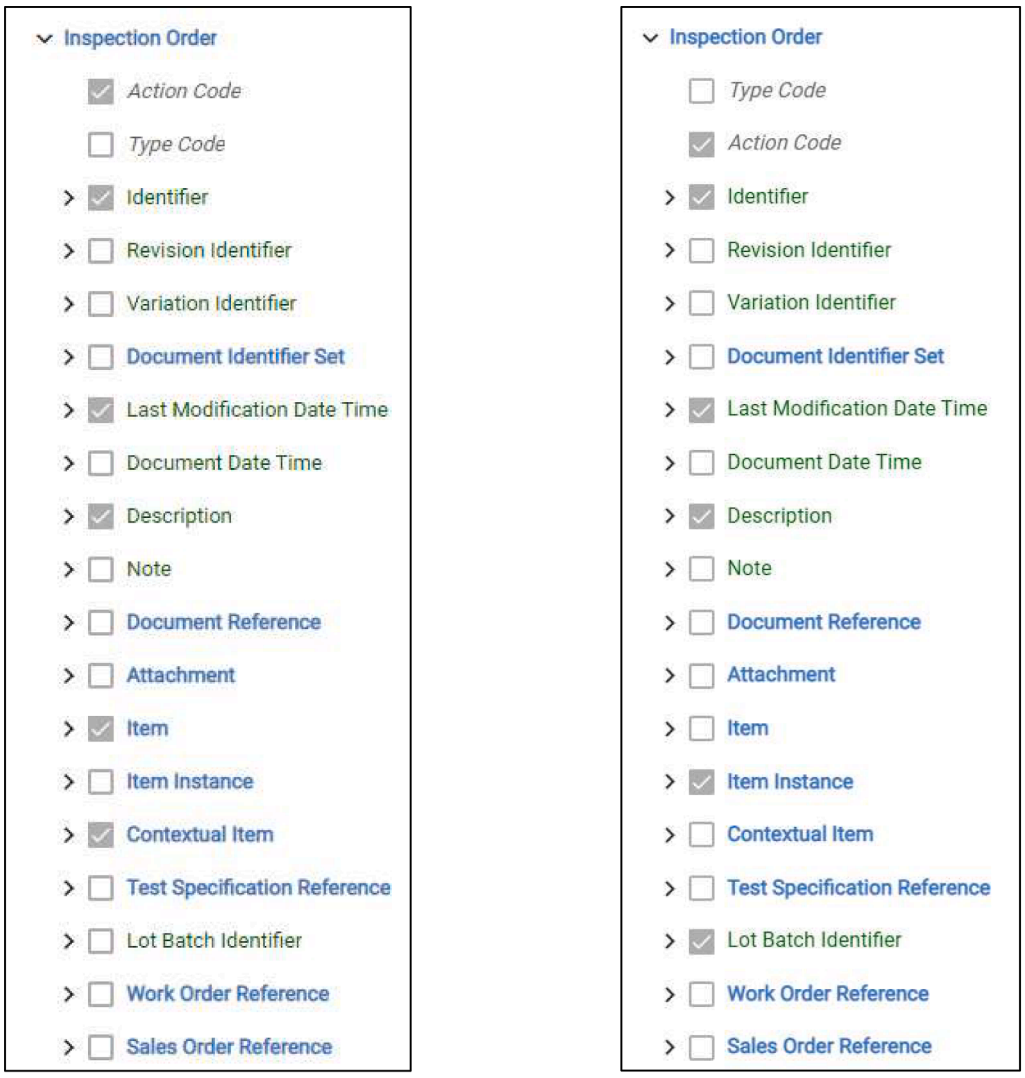
In CCS, each DES is represented in a model called a Core Component (CC). BIEs are then derived from these existing CCs; but they are associated with a specific Business Context (BC). A BC that indicates the situation(s) in which CCs are applicable. There are a few ways to represent a BC [15–17]. The simplest way is based on a combination of context values from various context schemes as illustrated in Fig. 3. The same figure illustrates the eight, context dimensions that drive the different BIEs from the same CC as stipulated by the CC specification.

Fig. 4 shows 1) various types of BIEs and CCs and 2) BIEs are derived from CCs (corresponding entities have a basis relation between them). Different types of CCs can be represented as a simple or complex, hierarchical, data structures where the complex ones can be composed of a collection of simple data structures. Specifically, ACC is a complex data structure, which comprises of ASCCs (a complex child structure) or BCCs (a simple child structure). ASCC structure reuses the ASCC Property, which is constructed from an optional qualification of the ACC with a property term.

For example, the 'Address' and 'Home Address' ASCC properties can be constructed from the ACC 'Address'. Similarly, a BCC reuses the BCC Property, which is constructed from a reusable Core Data Type (CDT) with an optional property called 'term qualification'. For example, the property term 'Tax' can be added to the CDT 'Amount. Type' to construct the 'Tax Amount' BCC Property. In addition to structural relationships between these CCs, CC entities have dictionary information such as Dictionary Entry Name (DEN) and Definition that can be used as semantic sources for schema matching. BIEs inherit such dictionary information from CCs and may add specifics as shown earlier in Fig. 2.

CDTs on the other hand have dictionary information (e.g., Data Type Term, Property Term in its Supplementary Components) plus value domains that contribute to the semantics of the BCC Property. Fig. 5 shows that the CDT consists of the CDT content component and the CDT supplementary component, each of which has a value domain represented by the Primitive Type. For example, the CDT 'Amount. Type' has the CDT content component 'Amount. Content' and the supplementary component 'Amount. Currency. Code'. The value domain of the content component is decimal, and that of the supplementary component is an international standard currency code list. The BDT inherits the CDT information. BDTs can restrict the value domains and specify the contextual definition.

Finally, we show an example instantiation of the CC model using the



(a) Raw Material Testing Business Context *(b) Finished Good Testing Business Context*

Fig. 1. Parts of two BIEs that are refined from the inspection order CC in different BCs of food manufacturing.

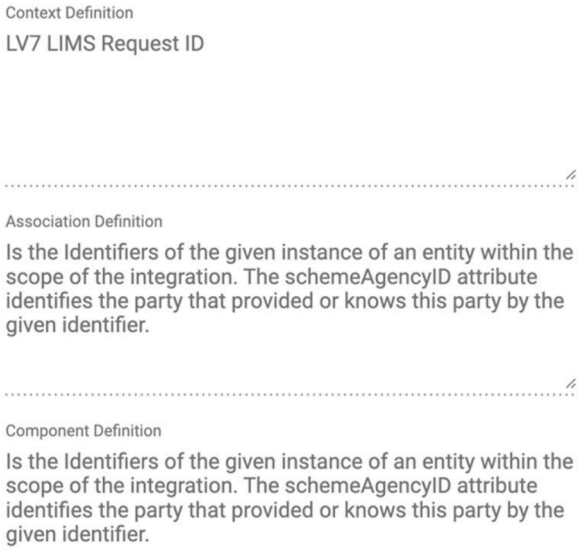


Fig. 2. Context definition of the field, identifier.

well-known purchase order document in Fig. 6. Dictionary Entry Names (DEN) are shown for each CC entity. BIEs derived from these CCs inherit these DENs and so as other information such as cardinalities and value domains, some of which may be restricted. Now, we turn to describing the BIE Uplifting functionality.

2.2. BIE uplifting

In the previous section, we described BIEs as profiles and restrictions of CCs. Both of which are present in any particular release of the DES. BIE Uplifting is a function that helps users transition their BIEs from being based on an older DES release to a newer one. It helps automate the migration process, which includes 1) determining where there are clear matches between data elements in the two releases, 2) providing suggestions when that is not the case, and 3) map and copy data from the old to the new entities.

However, implementing the BIE Uplifting process is complicated by two factors: BIE extensions and DES changes. BIE extensions are needed when additional data elements must be added in the current BIE. Such data elements arise for two reasons. First, when new requirements that are not supported by existing CCs are uncovered during BIE creation. Second, there are proprietary and specific data elements that companies do not want included in the standard.

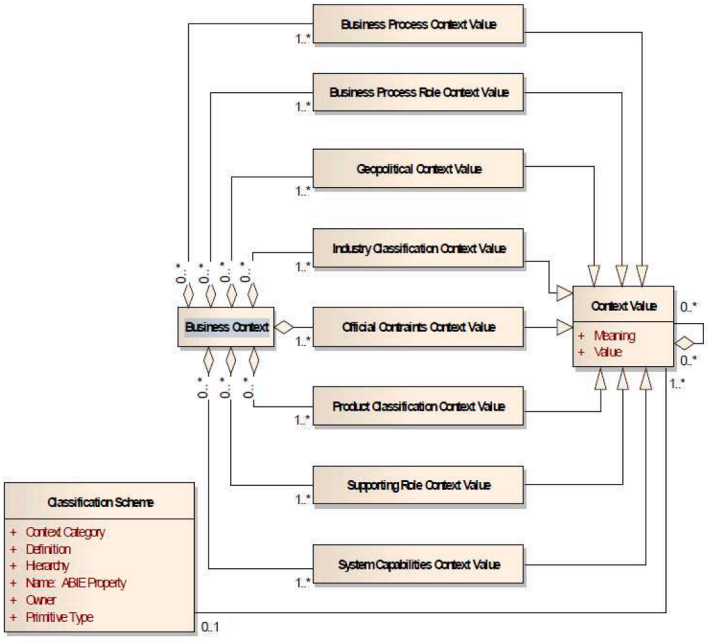


Fig. 3. Business context model according to CCS version 3.0 [9].

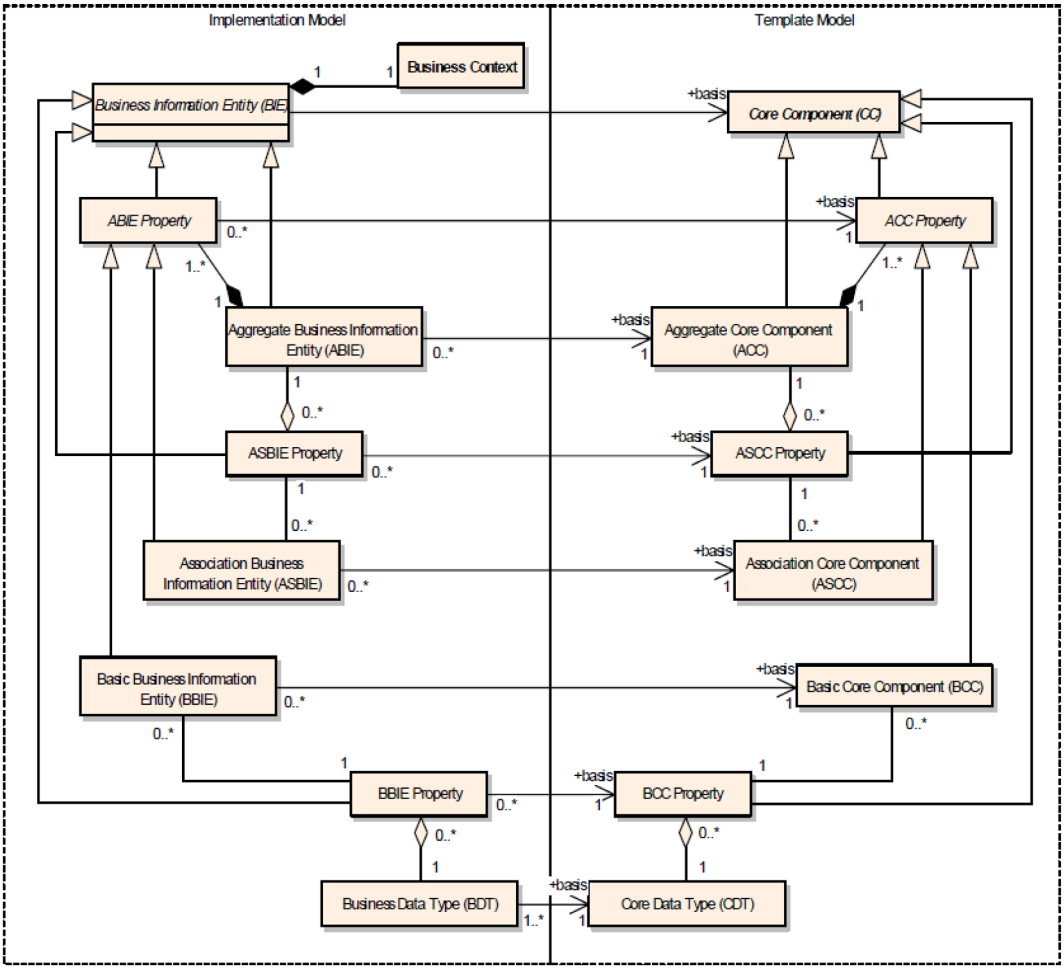


Fig. 4. Relationship between BIE (usage specification) and core component (DES) [9].

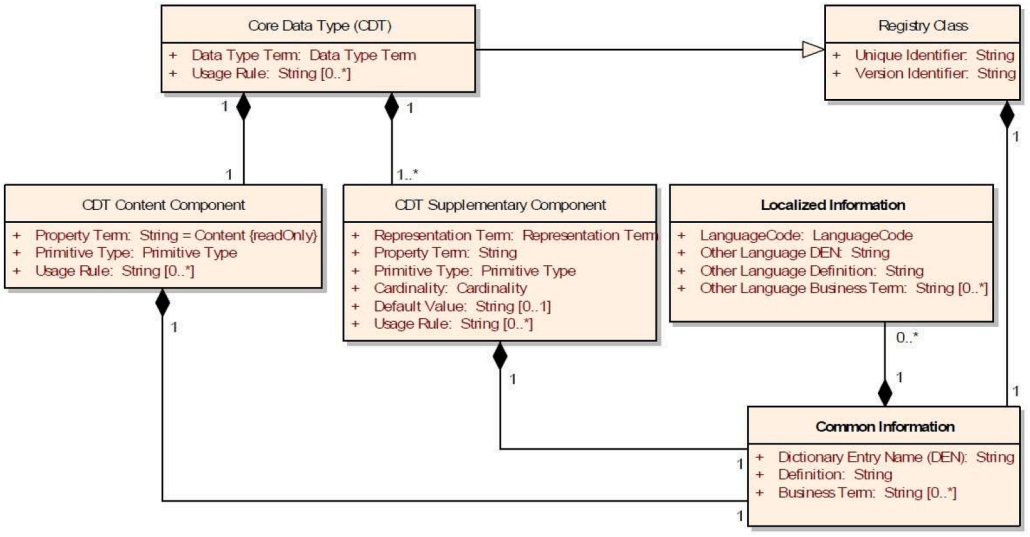


Fig. 5. Part of the core data type model from the CCS specification [9].

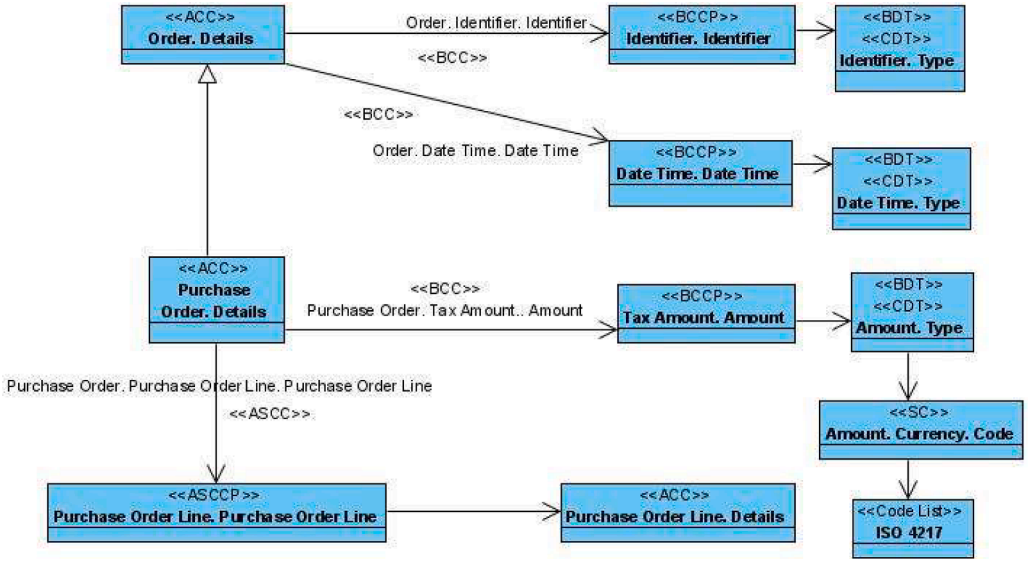


Fig. 6. Example instantiation of the CC model using the purchase order business document.

In the case of the former, there are two types of changes: major and minor. A major change occurs when the new DES is based on backward-incompatible changes. For that reason, DES typically does not introduce major changes until there are significant, architectural adaptations. Examples include changes in normative syntax, underlying design conventions (e.g., meta-model), or data typing. Currently, BIE Uplifting does not consider these types of major changes.

In the case of the latter, most minor changes are backward-compatible. Minor changes include 1) requirements introduced by new integration scenarios driven by BIE extensions, and changes in context dimensions such as those illustrated in Fig. 3; 2) bug fixes; 3) changes in an external source used by the DES. These changes could result in modifications in the data-element name, definition, structure, value domain, or deprecation. That final modification could lead to the eventual elimination of the data element.

Migrating a BIE from an old to a new DES release will be impacted by these changes. These impacts occur because they require standard users to map and adapt the affected, existing, data elements to the data elements in the new DES release. Since mapping and data restructuring is known to be a combinatorial problem, BIE Uplifting is intended to save users the time necessary for dealing with these complexities. It is also intended to enhance consistency and interoperability across BIEs sharing similar data elements. In this vein, schema matching can be beneficial to the BIE Uplifting function [18] and hence we turn to literature review on the topic.

3. Literature review

Most data exchange standards are based on information schemas. Analyzing standard DES for its development, adoption, and implementation requires matching those schemas. Numerous techniques have been developed for automatic (or semi-automatic) schema matching. Starting in the late 90s' and extending to the early 2000s', common approaches relied on various schema information such as element names, data types, structural properties [19–23] and the different characteristics of instance data [24–27]. However, in most cases instance data are not available. Combining individual techniques (hybrid approach) that produce one or more similarity measures were the most common approach to achieve higher schema-matching accuracy [28–35].

Techniques for using schema information to produce similarity measures can generally be classified into two categories - lexical-based or linguistic-based. Lexical-based similarity measures compare two pieces of schema information as if they themselves are the meanings [29,31]. Linguistic-based approaches on the other hands extract the meanings from schema information using NLP techniques first and then compare them [36,37]. However, traditional NLP techniques, such as a bag-of-words model, have difficulty capturing semantic information in DESes, especially for large and complex schemas with heterogeneous data structures. Additionally, as the sizes of schemas increase, more computational resources and time are typically required such that it becomes user-unfriendly and cost prohibitive to use.

Many NLP techniques use a vector-space model. The core advantage of representing words in a vector space is that vector operations can be used to solve language-processing tasks such as classification [38–40] and similarity matching [41,42]. However, earlier approaches in vector space models have limitations in dealing with semantic capturing. For instance, a bag-of-words model transforms given text documents into a set of words, including the number of occurrences. But it disregards other linguistic characteristics such as grammar or word order. Therefore, researchers increased the dimension of the vector space, (calling it "bag size") to try to increase accuracy. Unfortunately, while such high-dimensional spaces require expensive, computational resources, it can still show poor results [43].

To improve those results, NLP researchers have developed efficient methods to represent inputs inside a "dense vector space" that retains linguistic characteristics. Linear transformation techniques such as LDA (Linear Discriminant Analysis) and PCA (Principal Component Analysis) have been adopted for reducing the dimensionality of that "dense vector space" [43]. Furthermore, recent studies have shown new methods for analyzing a distributed representation of words, namely word embeddings, can provide significantly better performance than both LDA and PCA. These new methods rely on the application of a neural network [44–47].

Most word-embedding techniques share the same underlying idea, which is based on the concept popularized by Firth [48]. Instead of mapping words individually, his idea is to continuously take a sequence of words as an input and formalizing them into a vector space. Moreover, other studies show that different input types such as paragraphs, documents, and even structured, data types could be a source of embeddings [49–51]. The same idea can be applied to the schema-matching problem by identifying a candidate, data component using a set of nearby, data elements.

Recently, word-embedding approaches such as word2vec, GloVe, and BERT, and ELO [44–47,52] in NLP have gained attention. Moreover, the application of word-embedding for graph analysis, called graph embedding, has also been widely researched for distributed representations of graphs to encode the graph structure [53–55]. Because word-embedding techniques learn to represent words in a way that they have similar representations when their meanings are similar. In this way, word-embeddings with the semantic relationships can help schema matching algorithms to identify correspondences between data elements. Recent schema matching research have shown promising results using word embedding techniques. Several research reviewed in [56] used embedding techniques on external knowledge to support schema matching with varying results. The paper itself studies the effects of varying knowledge sources and usage strategies and concluded that the choice of strategy has a greater impact on the matching result. Koutras et al. [57] proposed to use graph embedding on column data to represent the column semantics. However, in many cases including the BIE Uplifting situation, data are not available to use in schema matching. In addition, the work did not consider the semantics inhered in relations/structure between tables. Hättasch et al. [58] commented that embedding based only on data could misguide the schema matching because data might be disjoint, but columns could still be semantically matched. They proposed a two-step embedding scheme that used, first, table and column information and then column data for matching. They demonstrated that the research showed promising results. Albeit it relies on data, which do not exist in our case. The approach also does not consider embedding based on relationship between table. Nevertheless, the table and column information embedding technique motivated us to further investigate word embedding for our problem context. That is, we would extend it to take into account relations. It should be noted that word-embedding techniques are attractive to our problem context from the computational perspective as well. Word embedding represents meaning of words in a lower dimensional space than other linear transformation methods, and similarities between all pairs of words in a given corpus can be precomputed. These computational characteristics make word-embedding techniques a promising linguistic-based similarity to be used in our user-interactive web-based application environment. Section 4.3 provides detail of our engineering and analysis to use word-embedding in a web server running on a small cloud compute without using instance data. It is used along with other lexical matchers in the overall hybrid scheme therefore we turn to that in the next section.

4. Hybrid-schema matching using word-embedding

Schema matching provides mapping suggestions by calculating similarities between components in two schemas. To simplify this calculation, we developed a 'hybrid-schema matcher' that aggregates individual matchers into a 'pipes-and-filters' architectural pattern. Each matcher has a scoring function with a range of [0, 1] that measures the similarity of the given, two input components passes the results to the next matcher through defined steps. All results obtained by the hybrid matcher are collected into a result set, which is then reported to the user. There are three matches in our hybrid matcher.

- Identifier Matcher: evaluates the similarity of two components using its unique identifier on the record.
- Field Matcher: evaluates the similarity between simple components (for example, a BCC Property) using a string-based, edit distance measure [59] based on their labels and value domain compatibilities.
- Structural Matcher: evaluates the similarity between aggregate components (for example, an ASCC Property) using word-embedding techniques.

Each matcher has a specific role. An identifier matcher reduces the size of inputs and computational complexity for other matchers. A field matcher and structural matcher are used when an identifier match is not applicable, for example when a component does not exist in a new standard release or vice versa. Field matcher is used for matching simple components, namely the BCCP. A structural matcher is for matching complex components, namely the ASCCP. In the following subsections, each match is described in more detail.

4.1. Identifier matching

Every component in SCORE is managed by a unique identifier, which is created using a pseudo-random, 128-bit number. Also, once it is assigned to the component, it will never change regardless of the number of future releases. It is conventional practice that standard developers do not change the semantics of any component such that the semantics is totally disjoint from its previous version once it was released. Thus, if two input components have the same identifier, they are semantically the same component - even if some of their properties are different. The identifier matcher compares identifiers between a source and a target component and returns 1 if they are the same otherwise returns 0.

Typically, due to interoperability concerns between various versions, components in a previous release are carried over to the next release through the lifecycle management process. Unless components are

marked deprecated in a new release, components with the same identifier can be matched with little computational expense by the identifier matcher. Thus, the matcher acts as an important filter to reduce the size of inputs and hence computational complexity in subsequent matchers.

4.2. Field matching

The BCCP functions, as a basic property of an ACC, represent a simple, business datum that has only one relationship with the primitive, core, data types - such as String Type, Integer Type, Boolean Type, etc. All objects, properties, data-type terms, and representation terms of components in CCS follow the very same, specific, naming rules [60]. These rules are used to represent the semantics of each component and to measure the string similarity between two different components. In other words, we could say that if two BCCP components have the same *property term* and point to the same *data type* they are semantically the same.

The string-similarity function attempts to quantify differences between two strings. Although numerous methods have been developed in this area, the most widely adopted methods for measuring string similarity are based on the 'edit distances', popularized by the Levenshtein distance [59]. It counts the number of operations (insertion, deletion, and/or substitution) required to transform one string into another. In our implementation, the Levenshtein result is normalized by the maximum length of given terms, $\max(|t_i|, |t_j|)$, where $|t|$ is a length of the term t. However, since the edit distance considers only character differences between two strings, it cannot capture other similarity characteristics in the term's value domain.

Consider the similarity between the BCCP, whose property term is 'Sequence Ordinal', and the set of BCCPs whose property terms are ['Sequence Code', 'Sequence Number'] as an example. If the similarity is based on the edit distance of the property terms only, the 'Sequence Code' shows a lower edit distance (i.e., higher similarity) than 'Sequence Number'. However, 'Sequence Number' should have higher similarity to 'Sequence Ordinal' because their value domains, respectively 'Ordinal' and 'Number', are more compatible than the domain between 'Ordinal' and 'Code'. Value domains of BCCPs are represented by their data-type terms according to CCS. For this reason, the result from the Levenshtein edit distance is enhanced with a data-type-term, similarity function, *DTT_score*.

DTT_score uses a hybrid approach combining measuring functions associated with 1) the data-type features such as allowed primitives, a default value domain and 2) the term similarity provided by WordNet [61]. First, BCCP records are categorized with 11 primitive types and 23 data type terms defined in core component data type catalog v3 definitions [62] as shown in Tables 1 and 2, respectively. Then *DTT_score* is computed as follows. Let t_i and t_j be data type terms for a given pair of two BCCPs. Then, the similarity function between the pair's data type terms is defined as

$$DTT\_score(t_i, t_j) = \begin{cases} 1.0 & , \ t_i = t_j \\ \tanh\left(\sum_{k\in\{p,\ d,\ s\}} \theta_k \cdot f_k\right) & , \ t_i \neq t_j \end{cases} \quad (1)$$

where f_k is an evaluation function for each factor including the allowable primitives (p), the default primitive (d), the term similarity based on WordNet (s), and θ is a coefficient for each function in the range (0, 1]. Precisely, f_p is a normalized, scoring function based on counting matched, allowed, primitive types between t_i and t_j. In a sense, f_d is a default-type indicator, which measures whether the default, primitive type is same or not, and f_s is a synonym indicator which measures whether each term is contained in the other's synonym list or not.

It is clear from the *DTT_score* definition that data-type terms will have a high similarity accuracy if they share allowable primitive types, a default value domain, and synonyms. For example, based on *DTT_score* function the data-type term 'Identifier' is like the data-type terms 'Name', 'Code', 'Text', 'Value', and 'Ratio' - but in descending order. It is important to note that all coefficients are adjustable in the application. Moreover, changing the value of the coefficients in *DTT_score* only affects the distribution of similarity values, but not the order of the listed results. As we can see in Fig. 7, extreme values of coefficients make skewed distribution. That is, a more balanced set of coefficients spread out the similarity values and could help the user discern the similarities better.

From our empirical assessment, the coefficients $\theta_p = 0.6$, $\theta_d = 0.4$, and $\theta_s = 0.3$ produce a good spread of similarity values for the OAGIS dataset in SCORE. Fig. 8 shows similarity values of the *DTT_score* for all data type terms with these coefficients. These coefficients are just default settings that can be adjusted by the user in the SCORE application. It should also be noted that Fig. 8 clusters data types that are known

Table 1
Primitive type definitions in CCS.

Primitive type	Description
Binary	Binary is a finite sequence of binary digits (bits).
Boolean	Boolean denotes a logical condition through a predefined enumeration of the literals true (The Boolean condition is satisfied) and false (The Boolean condition is not satisfied).
Decimal	Decimal is a subset of the real numbers, which can be represented by decimal numerals.
Double	Double is the IEEE double precision 64 bits floating point type.
Float	Float is the IEEE simple precision 32 bits floating point type.
Integer	Integer is a value in the infinite set (…−2, −1, 0, 1, 2…), a denumerably infinite list.
String	String is a sequence of characters in some suitable character set.
Normalized String	Normalized String is a string that does not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters.
Token	A token is a string that does not contain the line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20) and that have no internal sequences of two or more spaces.
Time Duration	Time Duration identifies a length of time in various time units as used in the Gregorian calendar: year, month, week, day, hour, minute, second, and fractions thereof.
Time Point	Time Point is a point in time to various common resolutions: year, month, week, day, hour, minute, second, and fractions thereof.

Table 2
Data type term mapping to value domain primitives in CCS data type catalog.

Data type term	Allowed primitives (Bold for default primitive)	Synonym terms
Amount	**Decimal**, Double, Float, Integer	Measure, Quantity
Binary Object	Binary	
Code	Normalized String, String, **Token**	
Date	Time Point	
Date Time	Time Point	
Time	Time Point	
Duration	Time Duration	
Graphic	Binary	Picture
Identifier	Normalized String, String, **Token**	
Indicator	Boolean	
Measure	**Decimal**, Double, Float, Integer	Amount, Quantity
Name	Normalized String, String, **Token**	Identifier
Number	**Decimal**, Double, Float, Integer	Amount
Ordinal	Integer	Number
Percent	**Decimal**, Double, Float, Integer	
Picture	Binary	Video
Quantity	**Decimal**, Double, Float, Integer	Measure, Amount
Rate	**Decimal**, Double, Float, Integer	Value
Ratio	**Decimal**, Double, Float, Integer, String	
Sound	Binary	
Text	Normalized String, **String**, Token	
Value	**Decimal**, Double, Float, Integer, Normalized String, String, Token	Measure, Rate
Video	Binary	Picture, Graphic

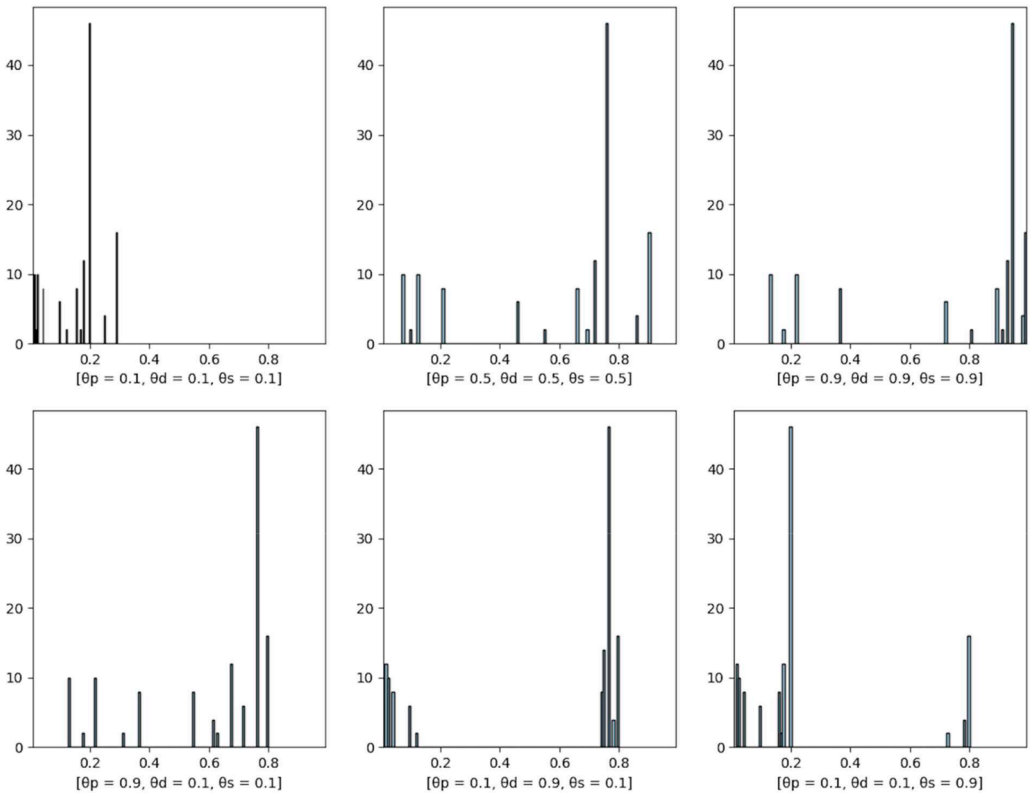


Fig. 7. Distribution frequencies of DTT_score values for different coefficients.

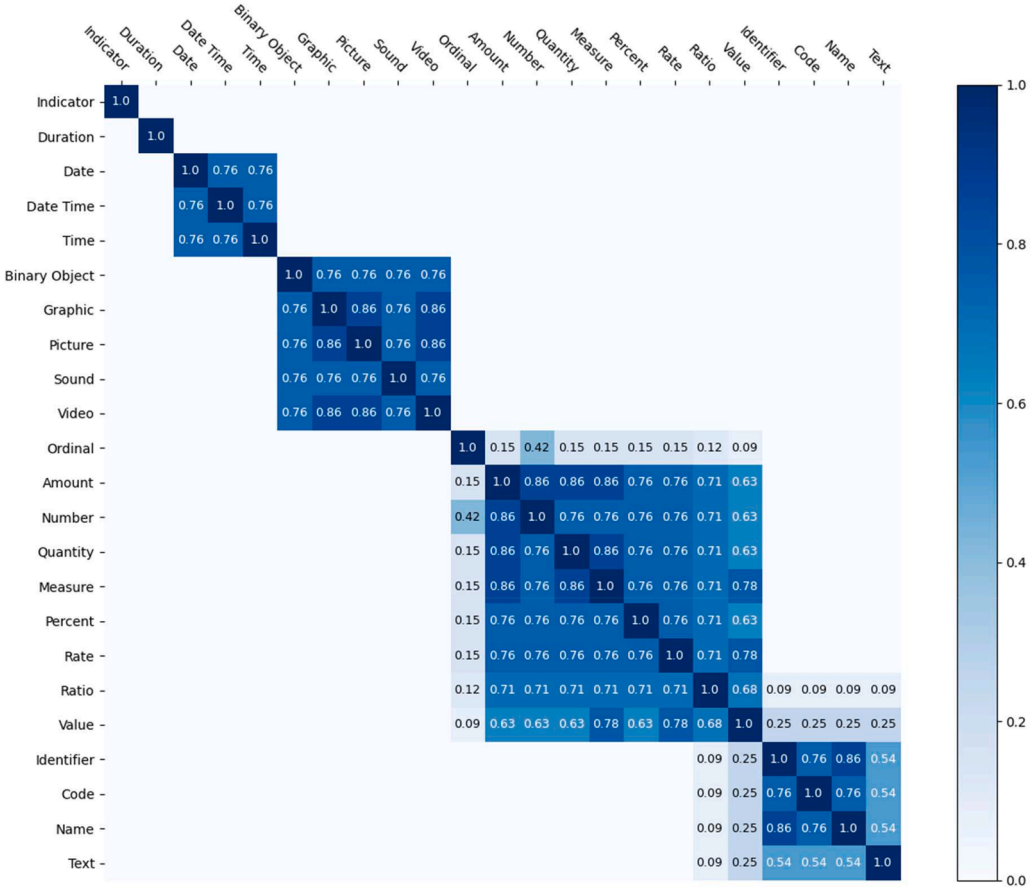


Fig. 8. Similarity values of the DTT_score for all data type terms with $\theta_p = 0.6$, $\theta_d = 0.4$, and $\theta_s = 0.3$.

to be semantically. The cluster shows relatively darker blues in the center of each cluster indicating that the similarity function does it work quite well.

Now, we can write the similarity measure for BCCPs (sim_{BCCP}) as follows,

$$sim_{BCCP}(b_i,\ b_j) \quad = w_d{\cdot}f_d(b_i[PT],\ b_j[PT]) \\ +(1-w_d){\cdot}DTT\_score(b_i[DT],\ b_j[DT]) \tag{2}$$

where $w_d \in [0,1]$ is a weight for the edit distance measure f_d, $b[PT]$ is the property term of BCCP, and $b[DT]$ is the data type term of BCCP.

To understand how much w_d affect the similarity for a string matching and a type matching, we analyze the quantitative relationship between the data type classification of BCCPs and sim_{BCCP} using F_1 score and $C(ed)$ over w_d as follows. F_1 score (Eq. (3)) is the harmonic mean of precision and recall using sim_{BCCP} matcher over the whole OAGIS BCCP corpus. The BCCP with the highest sim_{BCCP} is considered the recall for a particular BCCP.

$$F_1 = \frac{2{\cdot}precision{\cdot}recall}{precision + recall} \tag{3}$$

$C(ed)$ (Eq. (4)) is the average distance between sim_{BCCP} and edit distance matching results using a normalized distance between two elements in results $d(x,\ y)$,

$$C(ed) = \frac{\sum_{x,y} d(x,y)}{N},\ d(x,y) = 1 - \frac{|i_x - i_y|}{N} \tag{4}$$

where x and y are respectively the elements in the edit distance results and sim_{BCCP} results, i is an index number of the result array, and N is a total number of results (i.e., BCCPs).

Fig. 9 plots the F_1 score and the closeness $C(ed)$ over w_d. It shows that there are no significant changes in F_1 when w_d is less than 0.5. And in the same range, the $C(ed)$ indicates that sim_{BCCP} and edit distance provide some differing results. However, when w_d is getting larger than 0.6, sim_{BCCP} results quickly approach the edit distance results, and the F_1 score also reduces quickly, i.e., the matching quality quickly degrades. The objective is to have a sim_{BCCP} that can offer some alternative matches beyond just using the edit distance, because in some cases, considering data type term is a better match. This graph shows that a w_d between 0.1 and 0.5 can provide such sim_{BCCP} while the matching performance (F_1) does not significantly degrade.

4.3. Structural matching

In this section, we describe how word-embedding techniques are deployed in the SCORE environment to fit its requirements and constraints. First, SCORE is a web application. Most computing resources running web applications are not designed for machine learning. In particular, the SCORE tool is deployed with standard data (or in this case, a standard dataset). Although pre-trained models can be delivered to each system, each instance of deployment contains unique data that varies over time; and, hence, those models need to be retrained. In the case of SCORE, the locally unique dataset are the user-defined core components. Fortunately, the data size of these components is typically not big; and some content is typically harmonized into new releases of the standard. This characteristic should allow the model to be retrained in less than an hour.

Table 3 depicts an example relationship between the structure of 'Address' types in OAGIS 10.7 and the UBL 2.3 standards [63]. Even if the two 'Address' components are defined in different standards, they will share 1) properties with same names (e.g., both have 'Address Line' (1) and 'Street Name' (5)), and 2) semantically identical names (e.g., 'Postal Code'–'Postal Zone' (13) and 'Geographical Coordinate'–'Location Coordinate' (14)). Cascading down the 'Geographical Coordinate' and 'Location Coordinate' may be structurally similar as well and hence their similarity should also contribute to similarity of the 'Address' component. This suggests that when enough properties in two different components are shared, they can be thought of as semantically identical. Hence, word embedding techniques can measure the similarity between components via data-structure information. But only if we can characterize the relationship between data elements in the structure in the form of a sentence.

Moreover, the vector-space model should be able to deal with unknown words or phrases in the datasets. This should be possible without retraining because it is not necessary that the trained model be updated every time an end-user creates a new component with a new term. Lastly, the model should be able to deal with the semantically same components in multiple releases without using their unique identifiers. This can occur, for example, when an end user extends the current release by adding a 'Product' component, which later gets incorporated into the next release as a standardized component. In this case, two 'Product' components will have different component identifiers in different releases; but they should have very good similarity. In summary, the chosen embedding technique for BIE Uplifting must be able to

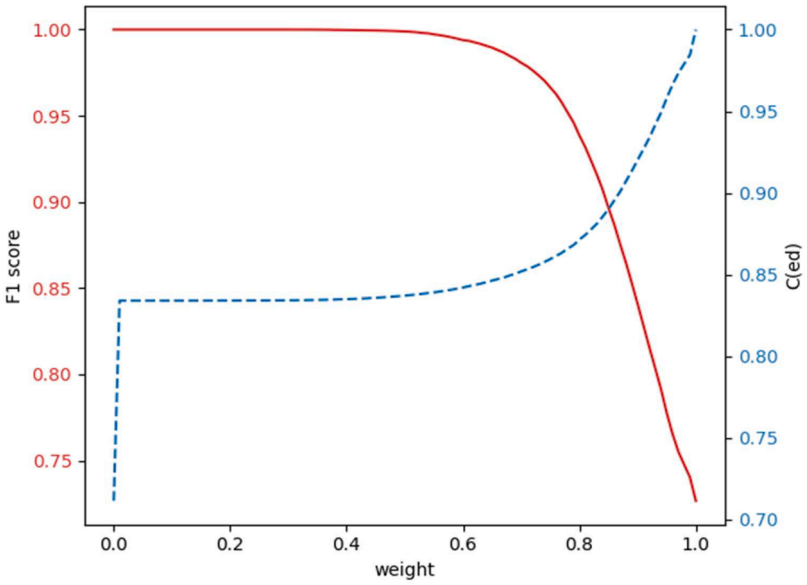


Fig. 9. A comparison of between F_1 score and the closeness $C(ed)$.

Table 3
The structure of 'Address' component in OAGIS 10.7 and UBL 2.3 standards [63].

[OAGIS] Address	[UBL] Address
Name	Address Type Code
Attention Of Name	Address Format Code
Care Of Name	Postbox (7)
Address Line (1)	Floor (6)
Building Number (2)	Room
Building Name (3)	Street Name (5)
Block Name (4)	Additional Street Name
Street Name (5)	Block Name (4)
Street Type Coordinate	Building Name (3)
Floor (6)	Building Number (2)
Unit	Description
Stair Case	Inhouse Mail
Door	Department
Post Office Box (7)	Mark Attention
Delivery Point ID	Mark Care
Plot ID (8)	Plot Identification (8)
City Name (9)	City Subdivision Name (10)
City Sub Division Name (10)	City Name (9)
Country Sub Division Code (11)	Postal Zone (13)
Country Code (12)	Country Subentity
Postal Code (13)	Country Subentity Code (11)
Status	Region
Preference	District
Geographical Coordinate (14)	Timezone Offset
Usage	Address Line (1)
	Country (12)
	Location Coordinate (14)

deal with limited computational resources and routinely changing data sets (sometimes every week).

A baseline technique of word embeddings in this study is word2vec that produces distributed representations in a vector space with one of the two neural network architectures: Continuous bag-of-words (CBOW) and Continuous skip-gram (shortly, Skip-gram) [52]. Fig. 10 illustrates CBOW and Skip-gram model architectures.

Both models are based on feedforward neural networks, but they have slightly different layer designs. CBOW takes surrounding words in an input layer to predict the current word. Skip-gram, on the other hand, predicts the context of the given current word to the output layer (See Fig. 10). Formally, the objective of CBOW and Skip-gram models for a given sequence of words in the corpus $w_1, w_2, \cdots, w_T$ is to maximize the following log-likelihoods:

$$\sum_{t}^{T} \log p\left(w_t \,\middle|\, w_{t-s}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+s} \right) [CBOW] \tag{5}$$

$$\sum_{t}^{T} \log p\left(w_{t-s}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+s} \,\middle|\, w_t \right) [Skip-gram] \tag{6}$$

where s is a size of the window (i.e., a length of the context words) surrounding the word w_t.

Word2vec has been shown to outperform other embedding techniques such as Latent Semantic Analysis (LSA) [39]. However, it still has the same problem: each word has a distinct, vector representation, but that representation ignores the semantics of that word.

FastText [64] is a recently developed method that extends word2vec model by using semantic information about the words. That information is based on the *n-grams technique*. In fastText, each word w_t is represented as a set of character n-grams and the word representation is substituted with a sum of other n-gram representations (i.e., $w_t = \sum_{g \in G_w} v_g$ where G_w is a set of n-grams appearing in the word w and v_g is a vector representation of the n-gram g). Specifically, each word is bracketed with special symbols '<' and '>' at the beginning and at the end. Then the word is decomposed into a set of n-grams. For example, the word 'Location' with the size of n-grams $n = 3$ would be a set of n-grams '<Lo', 'Loc', 'oca', 'cat', 'ati', 'tio', 'ion', 'on>', and '<Location>'. The original authors said that taking a range 3–6 for the size of n-grams provides a reasonable amount of sub-word information for both English and German datasets [64]. They also use an index of n-grams to bound the memory requirements. To generate the index, the variant Fowler-Noll-Vo hashing function is used to transform a set of n-grams to an integer value between 1 and $K = 2 \times 10^6$.

FastText model has two main advantages. First, it is interchangeable with other types of word2vec model, and the data format is compatible across all the subtypes. Thus, it brings flexibility when switching between different models. Second, fastText model handles out-of-vocabulary words better than word2vec model. Because it uses n-gram decomposition therefore there are possibilities that the same tokens are already presented in the trained model when it encounters the out-of-vocabulary words. One drawback is that more computational resources are needed during the training process since it uses granular tokens. However, this can be offset by a linear transformation to shrink

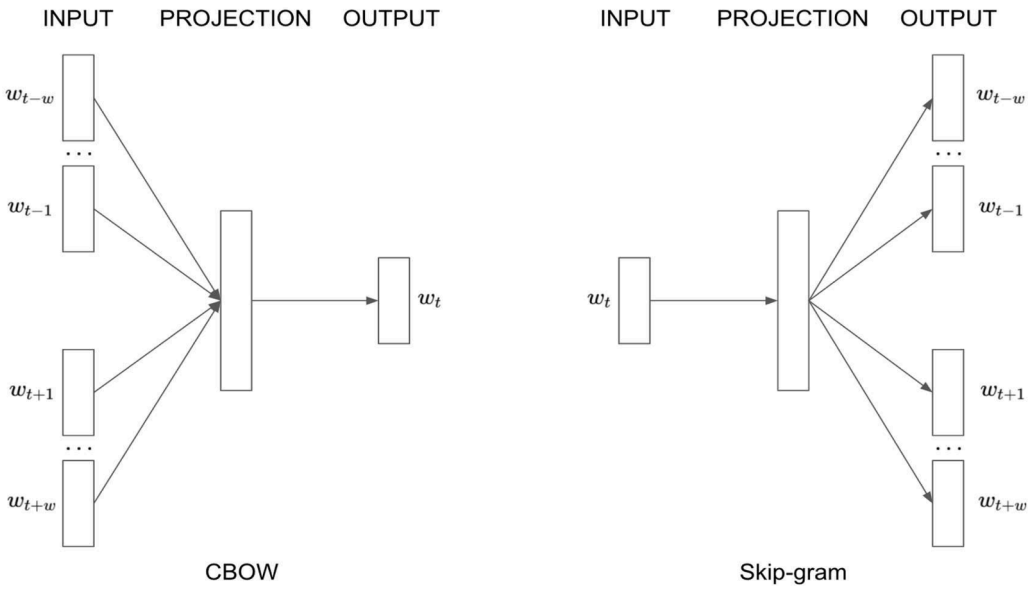


Fig. 10. CBOW and skip-gram models in word2vec [52].

the training corpus. Below we describe this approach.

Our approach to shrinking the training corpus is based on transforming the hierarchical structure of the complex component into a linear one and to eliminate duplicate series of terms in the sentence due to a circular reference (see the ASCC Property (ASCCP) example in Table 4). The transformation requires collecting all possible paths by traversing the ASCCP's descendant nodes using a depth-first search (DFS) approach. If the DFS encounters a circular reference, the DFS backtracks along its previous path until the last node before the circular reference is found. Moreover, the size of the corpus is influenced by the depth of paths, a.k.a., the sentence length. The discussion of the appropriate sentence length is provided below. Without a limit on the sentence length, a corpus size would incur significant computational cost with little gain on the matching performance.

Specifically, the potential, multiple transformations of an OAGIS dataset can result in approximately 3.6 billion sentences and 42.3 billion tokens. Table 4 illustrates an example of a dataset transformation into a training corpus for fastText using a 'Item' ASCCP with the maximum depth of the tree (i.e., a number of nodes from a root node to farthest leaf node) $d = 3$. Note that each term transformed from a dataset is concatenated with '_' (underscore) character for multi-words to make all terms unique in the corpus. Also, each term in each specific release is made unique by the release number suffix in a bracket pair. For example, 'Address[10.6]' and 'Address[10.7]' are treated as different terms in the corpus even if 'Address' component is the same through releases 10.6 and 10.7.

To evaluate which model shows better performance for the ASCCP matching task, we did a comparison between word2vec and fastText models. The comparison uses a precision measurement that counts the best-matched components in the target releases for all ASCCPs in the current release. In other words, it expects that the input 'Address[10.6]' will retrieve the best matched result 'Address[10.7]' in uplifting between 10.6 and 10.7 releases. Each testing was performed on six OAGIS releases from 10.6 to 10.7.4 using various values of parameters with a maximum depth d and a dimensionality of the word vectors v. The word-embedding process is also run ten times per test and the average precision value is then obtained. We take a range of 3 to 6 for the size of n-grams for fastText mode. Table 5 shows the comparison results.

The fastText approach significantly outperformed word2vec for all cases. The comparison stopped at $d = 5$ because (1) it took more than an hour to generate a dataset corpus and train the model in 8-core CPU environments, (2) the performance was already quite good at $d = 5$, and (3) the cost benefit of $d > 5$ both from the computation time (over one hour) and monetary cost perspectives was not worth it. The evaluation also showed that changing the vector size v did not meaningfully increase the performance.

Table 4
An example of a dataset transformation using a 'Item' sample data with $d = 3$. A circular reference case is marked in bold.

Item	"Item"
Bar Code ID	"Item Bar_Code_ID"
ID	"Item Bar_Code_ID ID"
Text	
Brand Name	"Item Brand_Name"
Text	"Item Brand_Name Text"
Disposition	"Item Disposition"
Location	"Item Disposition Location"
Coordinate	
Name	
Text	
Item	"Item Disposition Item"
Organization	"Item Disposition Organization"
Name	
Text	
Model Name	"Item Model_Name"
Text	"Item Model_Name Text"

Table 5
Comparison word2vec (WV) and fastText (fT) with CBOW (CB) and skip-gram (SG) models by evaluating ASCCP matching tasks with a maximum depth d and a vector size v. The best results are marked in bold for each setting.

		$v = 100$	$v = 200$	$v = 300$	$v = 400$	Avg.
$d = 3$	WV.SG	13.17 %	12.38 %	12.02 %	12.67 %	12.56 %
	WV.CB	12.20 %	12.07 %	11.46 %	10.89 %	11.66 %
	fT.SG	**68.35 %**	68.03 %	67.22 %	67.01 %	67.65 %
	fT.CB	50.48 %	49.39 %	48.45 %	49.58 %	49.48 %
$d = 4$	WV.SG	28.16 %	28.25 %	27.59 %	27.71 %	27.93 %
	WV.CB	19.04 %	19.50 %	18.83 %	18.97 %	19.09 %
	fT.SG	94.90 %	94.94 %	94.83 %	94.84 %	94.88 %
	fT.CB	98.01 %	**98.17 %**	98.02 %	97.91 %	98.03 %
$d = 5$	WV.SG	22.08 %	21.51 %	20.77 %	20.53 %	20.72 %
	WV.CB	14.57 %	14.89 %	14.64 %	14.61 %	14.68 %
	fT.SG	96.92 %	97.75 %	97.84 %	97.81 %	97.58 %
	fT.CB	98.18 %	98.34 %	**98.37 %**	98.41 %	98.33 %

5. Implementation

As described above, the schema-matching process needed for BIE Uplifting takes two BIEs from two different releases. The source BIE is from an older release while the target BIE is in a newer release. The user seeks to migrate the source BIE nodes and their associated data to their counterparts in the new release. For this to happen, the source (s) and target (t) BIEs must comply with the following rules:

Rule 1. The target release r_t must be published later than the source release r_s (i.e., $r_s < r_t$).

– If $r_s = r_t$, the process would be able to substitute with the copying. Otherwise, if $r_s > r_t$, some BIEs could break backward compatibility.

Rule 2. If BIE_s is based on CC_s in r_s, there must be a CC_t in r_t where $CC_s \cong CC_t$, such that $GUID(CC_s) = GUID(CC_t)$, where $GUID(x)$ is the globally unique identifier of the entity x.

– If *Rule 1* holds that is $r_s < r_t$, then *Rule 2* holds in the same SCORE instance as SCORE does not allow a CC to be discarded between releases due to the backward compatibility business rule.

Because of the compliance to *Rule 2*, CC_s and CC_t must be similar, typically CC_t tree is a superset of CC_s tree. This significantly reduces the Identifier Match processing time. The Field and the Structural Matchers address the components in the tree where the Identifier Matcher cannot match. To execute the Word Embedding Matcher in an instance of the SCORE tool, the instance needs to generate its own trained models for the source and the target releases (See Fig. 11).

All the parameters in Fig. 11 will affect model's training time and measurement performance. The default values for all parameters provided relatively good, matching performance (over 90 % correct matches in the experiments) with model generation time of less than an hour using a general-purpose Amazon EC2 computer that is affordable for small companies such as OAGi.

The BIE Uplifting user interface (UI) also has configurable preferences including 1) the suggestion threshold and 2) the switch to include or exclude ASCCP or BCCP in the result set. These configurations allow for more focused display of the matched results and more efficient uplifting. In particular, the suggestion threshold limits the results with a similarity value greater than or equal to the threshold value. The ASCCP and BCCP switch allows the user to include or exclude the ASCCP or BCCP from the result set based on the user's interest.

Fig. 12 shows the BIE Uplifting UI of the source and target BIE. Components that are successfully matched by the Identifier Match do not have a selection box and they do not need any user action. When the user clicks a component that has a selection box in the Source BIE Tree pane on the left, suggestions for mapping based on the schema-matching algorithm show up in the Suggests box on the right most side in Fig. 12.

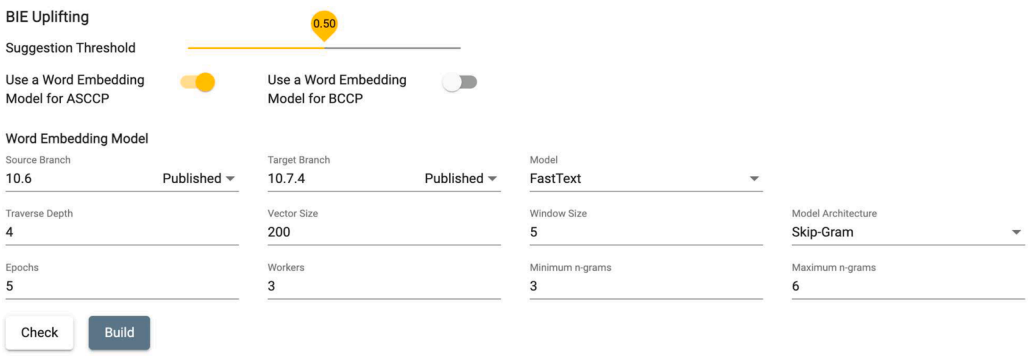


Fig. 11. UI for generating word embedding models for the BIE uplifting.

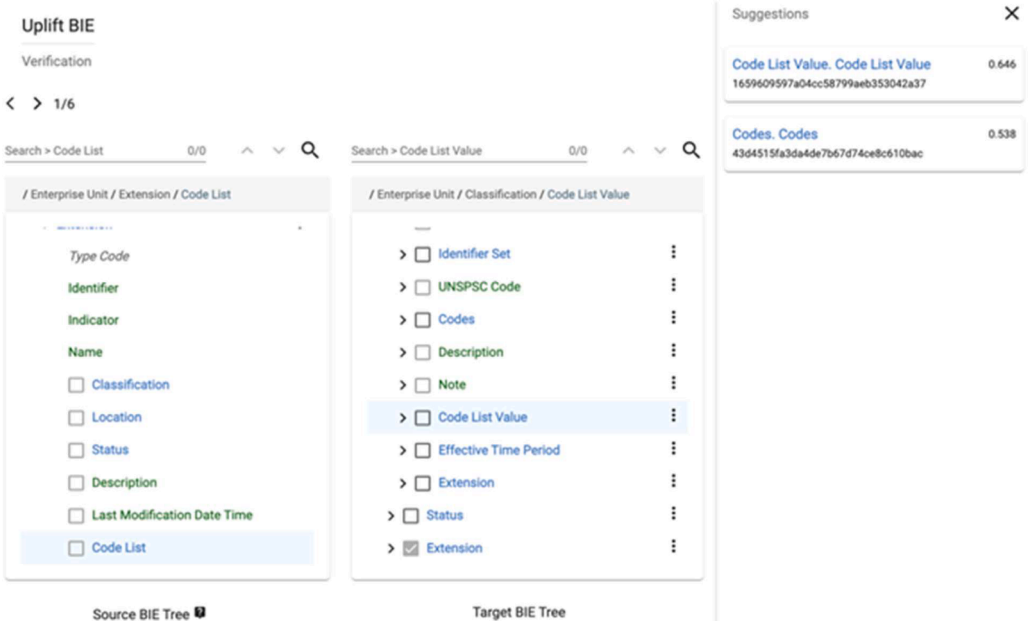


Fig. 12. BIE uplifting UI showing the source and target BIE tree and the suggested uplift target (top right).

Furthermore, if the user clicks on one of the suggestions, the corresponding component will be displayed in the Target BIE Tree pane. For example, in Fig. 12, the *Code List* component is selected in the Source BIE Tree; *Code List Value* and *Codes* components are displayed in the Suggestions box; and the user selects the *Code List Value* suggestion, so the *Code List Value* node is highlighted in the Target BIE Tree pane. The user can investigate details of the source and target nodes by expanding the tree and opening the detail page of each node.

Once all the nodes in the source BIE tree are mapped to the nodes in the target BIE tree using the matching suggestions, the user specific implementation details in the source nodes are automatically transferred to corresponding target nodes. In other words, the user does not have to create usage specifications manually for the new version of the standard. Currently, generating the data migration code is outside the scope of SCORE.

6. Summary

The SCORE tool provides advanced methodologies and software applications for data exchange standards (DES) management, which includes the functionalities needed for standard developers and users. Without such advancements, standard development, use, and maintenance would require significantly more effort, cost, and time. Moreover, it could instead take years for business and manufacturing organizations to achieve the cost savings and business agilities they expected from deploying both Industrie 4.0 and Smart Manufacturing.

Over the past seven years, the Systems Integration division at NIST has led a collaboration with industry members in the OAGi Message Standard Semantic Refinement Methods and Tools working group (recently renamed the SCORE working group). The group's goal was to develop a new framework, which includes new practices and tools for efficient DES development and use. That framework had led to a new practice for more efficient DES management, namely the SCORE tool [5].

SCORE has been used by the OAGi standard consortium to significantly reduce the time needed to develop and update its OAGIS DES - from between 6 and 12 months to every month. While this is good news, standard users also must deal with transitioning their usage specifications, which are based on an older release of the standard, to be based on a new release more often. As part of the SCORE tool, the BIE Uplifting function was developed to help standard users make that transition. Without BIE Uplifting, companies would usually shy away from keeping their integration ecosystem up to date with new standard releases. This would cause interoperability issues over time because 1) different companies in the supply chain use different releases and 2) different departments within the same company keep extending the standard in varying ways. Because BIE Uplifting has combinatorial complexity due to the mapping task, this paper investigated a modern, AI- and machine

learning-based, schema matching to further assist the user.

Apart from introducing a new framework for managing DES and its uses, the contribution of this paper includes an explanation of the various types of changes in the DES. Changes that users had to deal without our BIE Uplifting and without our hybrid-schema matcher that considers several semantic elements. Examples include identifier, data-element name, value domains, and structure. Additionally, we presented a field-matching similarity, which was further decomposed into lexical matching and value domain matching. Approaches to coming up with weights and coefficients across these similarity functions were presented. In particular, the value domain (data type term) similarity was based on an activation function commonly used in natural language processing. It uses the data type name, WordNet, and primitives to calculate the similarity between different simple contents of data elements. An analysis showed an interesting characteristic of the value domain similarity. That is, while a more evenly distributed coefficients for each of the field-matching component can make the overall similarity more discernable, they are insensitive to the similarity ranking. Selecting coefficients is a typical issue in a composite similarity measure. The analysis showed that using such a function in a composite similarity function can alleviate such issue. The field matcher and the identifier matcher are connected in a sequential pipeline to reduce the problem size for a more complex situation that needs structural consideration. For that, a structural similarity using two word-embedding techniques, namely word2vec and fastText, on a tree data structure were investigated due to their potential computational benefits when compared to other traditional linear transformation NLP techniques. The paper shows the engineering of word2vec and fastText to fit the hierarchical data and the computational constraint of web-based applications. It includes a depth-first tree traversal transformation logic to convert the hierarchical data into sentences, an algorithm to shrink the corpus, and an analysis to figure the appropriate depth and vector size. For our dataset, the research showed that fastText outperformed word2vec, while increasing the vector size over 100 does not give significant gain and the maximum depth of 4 or 5 gives reasonably good matching performance at over 98 %. Although recently proposed word-embedding techniques, such as BERT, may perform better for mappings, our study showed that using fastText with these parameter values still provides acceptable performance while using affordable computational time and resources. Since we took into account the feasibility in terms of workloads in business systems, the affordability is based on the budget available to the typically small, SDOs and the desired, user-response time from their users.

7. Future work

The machine-learning-based BIE Uplifting showed promising results based on industry feedback. Two comments were provided that include 1) the capability would also be useful for standard mapping tasks, and 2) context definitions and deprecation information should be considered in the matchers in addition to data element labels and data types. Part of our research is to address the latter comment. It will involve understanding the trade-off between matching performance and the additional computational resources needed to incorporate the required, additional data.

While SCORE has been used successfully to manage the OAGIS standard development and use, additional work is needed to incorporate schema matching to be used by industry practitioners particularly from the cost vs. value proposition perspective. One of the future works will focus on the ability to host and handle other standards that use different naming rules and usages. Industry users would like to maintain mappings between their data definitions and these evolving, data standards. There are also other growing industry needs for 1) a computer-aided, semantic, gap analysis between an (for example) an enterprise-level, application interface definition and the target DES and 2) the semantic matching between enterprise-level-application vocabulary (business term) or another DES and the target DES. We believe schema matchers will offer a great value proposition to them if ease of maintenance and system responsiveness are maintained. However, it will be much more computationally intensive to match between two different standards or between an application interface definition and a standard. This will happen because the identifier matcher cannot be used to filter the tree for the field and structural matchers. Furthermore, while exploring other embedding technique such as graph embedding could be another interesting dimension to explore, developing a strategy where computational cost and time are acceptable to the user and organization will be a challenge as additional steps are needed.

CRediT authorship contribution statement

Hakju Oh: Conceptualization, Methodology, Software. **Boonserm (Serm) Kulvatunyou:** Resources, Data curation, Writing – original draft. **Albert Jones:** Writing – review & editing. **Tim Finin:** Writing – review & editing.

Declaration of competing interest

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

Data availability

The data that has been used is confidential.

Acknowledgment

Certain commercial systems and applications identified in this paper are not intended to imply recommendation or endorsement by the National Institute of Standards and Technologies, nor is it intended to imply that they are necessarily the best available for the purpose. This work is funded by NIST collaborative research agreement 70NANB20H029.

References

[1] K.-D. Thoben, S. Wiesner, T. Wuest, Industrie 4.0" and smart manufacturing-a review of research issues and application examples, Int. J. Autom. Technol. 11 (1) (2017) 4–16.
[2] A. Kusiak, Smart manufacturing, Int. J. Prod. Res. 56 (1–2) (2018) 508–517.
[3] S. Mittal, M.A. Khan, D. Romero, T. Wuest, Smart manufacturing: characteristics, technologies and enabling factors, Proc. Inst. Mech. Eng., Part B: J. Eng. Manuf 233 (2019) 1342–1361.
[4] S. Vaidya, P. Ambad, S. Bhosle, Industry 4.0 - a glimpse, Procedia Manuf. 20 (2018) 233–238.
[5] B. Kulvatunyou, H. Oh, N. Ivezic, S.T. Nieman, Standards-based semantic integration of manufacturing information: past, present, and future, J. Manuf. Syst. 52 (2019) 184–197.
[6] I. Grangel-González, P. Baptista, L. Halilaj, S. Lohmann, M.-E. Vidal, C. Mader, S. Auer, in: The industry 4.0 standards landscape from a semantic integration perspective, ETFA, IEEE, 2017, pp. 1–8.
[7] Lu, Y., Morris, K.C., & Frechette, S., (2016). Current standards landscape for smart manufacturing systems, National Institute of Standards and Technology, NISTIR, 8107, 39.
[8] S. Weyer, M. Schmitt, M. Ohmer, D. Gorecky, Towards industry 4.0-standardization as the crucial challenge for highly modular, multi-vendor production systems, Ifac-Papersonline 48 (3) (2015) 579–584.
[9] UNECE, (2009). UN/CEFACT core components technical specification version 3.0.
[10] ISO-15000-5:2014, (2014). *ISO 15000-5:2014 electronic business extensible markup language (ebXML) — part 5: core components specification (CCS).* Retrieved from https://www.iso.org/standard/61433.html.
[11] OAGi, The open application group integration specification (OAGIS), https://oagi.org/pages/connectspec.
[12] AgGateway. ADAPT (inter-operability), https://www.aggateway.org/GetConnected/ADAPT(inter-operability).aspx.
[13] AgGateway, (2019, December). AgGateway 2019 December newsletter, https://www.aggateway.org/News/Newsletter/2019Newsletters/2019DecemberNewsletter.aspx.
[14] G.A. Gibbon, A brief history of LIMS, Lab. Autom. Inf. Manag. 32 (1) (1996) 1–5.

[15] D. Novakovic, C. Huemer, Business context sensitive business documents: business context aware core components modeling using the E-UCM model, in: The 11th IEEE International Conference on Industrial Informatics (INDIN), 2013, pp. 523–528.
[16] E. Jelisic, N. Ivezic, B. Kulvatunyou, P. Milosevic, S. Babarogic, Z. Marjanovic, A novel business context-based approach for improved standards-based systems integration—a feasibility study, J. Ind. Inf. Integr. 30 (2022) 100385.
[17] Unified context methodology technical specification, (2010). https://studylib.net/doc/7241720/un-cefact-unified-context-methodology-technical (accessed December 28, 2022).
[18] E. Rahm, E. Rahm, Towards large-scale schema and ontology matching, in: Z. Bellahsene, A. Bonifati (Eds.), Schema Matching and Mapping, Springer-Verlag, Berlin Heidelberg, 2011, pp. 3–27.
[19] S. Bergamaschi, S. Castano, M. Vincini, D. Beneventano, Semantic integration of heterogeneous information sources, Data Knowl. Eng. 36 (3) (2001) 215–249.
[20] W.S. Li, C. Clifton, SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks, Data Knowl. Eng. 33 (2000) 49–84.
[21] J. Madhavan, P.A. Bernstein, E. Rahm, Generic schema matching with cupid, VLDB (2001) 49–58.
[22] T. Milo, S. Zohar, Using schema matching to simplify heterogeneous data translation, VLDB (1998) Z24–Z27.
[23] L. Palopoli, G. Terracina, D. Ursino, The system DIKE: towards the semi-automatic synthesis of cooperative information systems and data warehouses, in: ADBIS-DASFAA Symposium, 2000, pp. 108–117.
[24] J. Berlin, A. Motro, Database schema matching using machine learning with feature selection, in: Advanced Information Systems Engineering: 14th International Conference, Springer Berlin Heidelberg, 2002, pp. 452–466. CAiSE 2002 Toronto, Canada, May 27–31, 2002 Proceedings 14.
[25] A. Doan, P. Domingos, A.Y. Halevy, Reconciling schemas of disparate data sources: a machine-learning approach, in: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, 2001, pp. 509–520.
[26] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Learning to map between ontologies on the semantic web, in: Proceedings of the 11th International Conference on World Wide Web, 2002, pp. 662–673.
[27] D.W. Embley, D. Jackman, L. Xu, Multifaceted exploitation of metadata for attribute match discovery in information integration, in: Workshop on Information Integration on the Web, 2001, pp. 110–117.
[28] D. Aumueller, H.-H. Do, S. Massmann, E. Rahm, Schema and ontology matching with COMA++, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005, pp. 906–908.
[29] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, S. Fienberg, Adaptive name matching in information integration, IEEE Intell. Syst. 18 (5) (2003) 16–23.
[30] A. Bonifati, E. Rahm, Schema Matching and Mapping, Springer Berlin Heidelberg, 2011.
[31] Cohen, W., Ravikumar, P., & Fienberg, S., (2003). A comparison of string metrics for matching names and records, Kdd workshop on data cleaning and object consolidation, pp. 73–78.
[32] H.-H. Do, E. Rahm, COMA—a system for flexible combination of schema matching approaches, in: VLDB'02: Proceedings of the 28th International Conference on Very Large Databases, Elsevier, 2002, pp. 610–621.
[33] F. Giunchiglia, M. Yatskevich, P. Avesani, P. Shvaiko, Semantic matching: algorithms and implementation. Journal On Data Semantics IX, Springer Berlin Heidelberg, 2007, pp. 1–38.
[34] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: Proceedings 18th International Conference on Data Engineering, IEEE, 2002, pp. 117–128.
[35] P. Shvaiko, J. Euzenat, A survey of schema-based matching approaches. Journal On Data Semantics IV, Springer Berlin Heidelberg, 2005, pp. 146–171.
[36] F. Giunchiglia, M. Yatskevich, P. Avesani, P. Shivaiko, A large-scale dataset for the evaluation of ontology matching systems, Knowl. Eng. Rev. 24 (2) (2009) 137–157.
[37] P. Jain, P. Hitzler, A.P. Sheth, K. Verma, P.Z. Yeh, Ontology alignment for linked open data, in: International Semantic Web Conference, Springer, 2010, pp. 402–417.
[38] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[39] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inf. Sci. 41 (6) (1990) 391–407.
[40] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.
[41] T.K. Landauer, P.W. Foltz, D. Laham, An introduction to latent semantic analysis, Discourse Process. 25 (2–3) (1998) 259–284.
[42] P.D. Turney, P. Pantel, From frequency to meaning: vector space models of semantics, J. Artif. Intell. Res. 37 (2010) 141–188.
[43] L. van der Maaten, E. Postma, J van den Herik, Dimensionality reduction: a comparative, J. Mach. Learn. Res. 10 (2009) 13.
[44] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K., (2018). BERT: pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv: 1810.04805*.
[45] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Adv. Neural Inf. Process. Syst. (2013) 26.
[46] J. Pennington, R. Socher, C.D. Manning, GloVe: global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[47] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L., (2018). Deep contextualized word representation, *arXiv preprint arXiv:1802.05365*.
[48] J.R. Firth, A synopsis of linguistic theory. Studies in Linguistic Analysis, 1957.
[49] A. Grover, J. Leskovec, node2vec: scalable feature learning for networks, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864, gRPC. (2016). *gRPC*. Retrieved from, https://grpc.io/.
[50] Q. Le, T. Mikolov, Distributed representations of sentences and documents, Int. Conf. Mach. Learn. (2014) 1188–1196.
[51] Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., & Jaiswal, S., (2017). graph2vec: learning distributed representations of graphs, *arXiv preprint arXiv:1707.05005*.
[52] Mikolov, T., Chen, K., Corrado, G., & Dean, J., (2013). Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.
[53] S. Cao, W. Lu, Q. Xu, Deep neural networks for learning graph representations, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2016, p. 30.
[54] Kipf, T.N., & Welling, M., (2016). Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907*.
[55] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.
[56] J. Portisch, M. Hladik, H. Paulheim, Background knowledge in schema matching: strategy vs. data, in: International Semantic Web Conference, Cham, Springer International Publishing, 2021, pp. 287–303.
[57] C. Koutras, M. Fragkoulis, A. Katsifodimos, C. Lofi, REMA: graph embeddings-based relational schema matching, in: EDBT/ICDT Workshops, 2020.
[58] Hättasch, B., Truong-Ngoc, M., Schmidt, A., & Binnig, C., (2022). It's AI match: a two-step approach for schema matching using embeddings, arXiv preprint arXiv: 2203.04366.
[59] V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Soviet Phys. Dokl. 10 (1966) 707–710.
[60] ISO/IEC-11179-5, (2005). ISO/IEC 11179-5 metadata registries (MDR) - part 5: naming and identification principles, Retrieved from https://www.iso.org/standard/35347.html.
[61] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[62] UNECE, (2011). UN/CEFACT core components data type catalogue version 3.1.
[63] Holman, K., (2021, January 19). Universal business language version 2.3, committee specification 01, Retrieved from Committee Specification 01: https://docs.oasis-open.org/ubl/UBL-2.3.html.
[64] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.