RESEARCH

A sample article title

Jane E. Doe[1*] and John R.S. Smith[1,2]

*Correspondence: jane.e.doe@cambridge.co.uk
[1]Department of Science, University of Cambridge, London, UK
Full list of author information is available at the end of the article

Abstract

First part title: Text for this section.

Second part title: Text for this section.

Keywords: sample; article; author

Introduction

Data and Information modeling in the healthcare domain have witnessed significant improvements in the last decade owing to advances in the development of state-of-the-art information and communication technologies (ICT) and formalization of storage and messaging standards. Subsequently, the scope of Healthcare Management Information Systems (HMIS), medical ontologies, and Clinical Decision Support Systems (CDSS) has broadened, beyond the operational capabilities of traditional rule based systems. One of the major reasons behind this limitation is due to the numerous heterogeneities in healthcare at data, knowledge, and process level. Thus, healthcare interoperability which aims to provide a solution to this problem, can be compartmentalized into data interoperability, process interoperability, and knowledge interoperability. Data interoperability resolves the heterogeneity between data artificats, to enable, seamless and interpretable communcation among source and target organizations, while preserving the data's original intention during storage, communication, and usage (as defined by IEEE 610.12 [1], Health Level Seven International HL7, and Healthcare Information and Management Systems Society HIMSS [2]). On the other hand, process interoperability regulates the communication among organizational processes to provide compatability between process artifacts within and seamless transformations across different organizations[3]. Lastly, knowledge interoperability provides a sharing mechanism for reusing interpretable medical knowledge, acquired through expert intervention and other mechanisms, across decision support systems [4]. In more tangible terms, healthcare interoperability at data, process, and knowledge level can be exemplified within the healthcare constraints experienced due to the emergence of Covid 19. The operational capabilities of the current healthcare service delivery infrastructure has gone under tremendous stress due to Covid 19. World over, large primary healthcare units have managed to create separate units for managing patients, suffering from extreme cases of the novel coronavirus. For secondary and tertiary care units, government involvement has become necessary to filter coronavirus patients and adhering to a national pandemic response policy. These complex circumstances have enhanced the need for sharing patient data and state-of-the-art medical knowledge in real-time, to provide the medical experts with a tool to make accurate and timely decisions. Data interoperability can enable the front line medical workers to fetch, understand, and use patient data, especially comorbidities, across organizational and physical boundaries, without suffering from societal taboos that may prevent the patient from sharing their complete and accurate medical histories. Knowledge interoperability can improve the knowledge acquisition and sharing protocols to provide the medical experts such as epidieomologists and vaccinologist, with latest information on affected population trends, disease diagnosis, treatment, and followup procedures, and interpretable decisions leading to positive or negative outcomes. Process interoperability can help reduce and in some cases remove the operational redundancies between health centers. In this way, successive healthcare treatments can take benefit from earlier diagnosis, treatment, and followup procedures, thereby reducing the stress on healthcare experts and systems. Standards such as Health Level Seven (HL7) Fast Healthcare Interoperability Resources (FHIR), and openEHR provide the foundations for storing and communicating medical data, through the use of well defined protocols. While systematized nomenclature of medicine—clinical terms (Snomed-CT) [5] and logical observation identifiers names and codes (LOINC) [6] provide a standard definition for clinical terminologies and laboratory tests, respectively. Similarly Medical Logic Module (MLM) provides a standardized way for expressing medical knowledge. However, the plethora of standards, necessitates the creation of bridging standards, that can resolve the heterogeniety between the medical standards. Substantial effort has gone into this endeavor with the Clinical Information Modeling Initiative (CIMI) [7] taking the lead in bridging the gap between HL7v3 and openEHR. Similarly, SNOMED CT and LOINC are working to resolve the redundancies between the two terminological standards since 2013. This healthcare interoperability solution follows a formal, albiet long process, which is greatly dependent on the human factor. However, the current healthcare scenario, requires a quick solution to create a scaffolding of an interoperable bridge between various healthcare providers. It is also important to ensure that this scaffolding should be able to support the formal standardization processes of the future. In [8]we have presented the Ubiquitous Health Platform (UHP), which provides semantic reconciliation-on-read based data curation for resolving data interoperability between various schema. This methodology is based on the creation and management of schema maps, that can provide the framework for transforming a source schema into a target schema. In the current manuscript, we will present our research work to build and manage the schema map knowledge base. Overall, our methodology has two major portions, firstly we apply a novel schema matching technique to create a transformation function σ for the participating legacy schema, and secondly, we have used the Ripple Down Rules (RDR) to manage our knwoledgebase, which will be presented in some detail, focusing primarily on the search and evolution services. In particular,

- Section 2 contains the details of our methodology, where we aim to present a reproducible theoretical framework.
- Section 3 provides the experimental setup
- Section 4 presents the results
- Section presents the related work
- Section 6 concludes the paper.

Related Work

Althubait et al. [9] proposed an ontology expansion methodology that identifies and extracts new class from text articles using word embedding and machine learning techniques. The authors identified the similarity of tokens and phrases of the text articles with the exiting classes of the ontology. The target ontology is expanded with classes from text articles having greater similarity with that of already added classes. A similar word embedding technique was also used by Nozaki et al. [10], where the authors used instance based schema matching technique to identify the semantic similarity between two instances. The results of the study showed the possibility of detecting similar string attributes of different schemas. Yousfi et al. [11] also utilized semantic base techniques and proposed xMatcher XML schemas matching approach. xMatcher transforms schemas into a set of words, followed by measuring words context, and relatedness score using WordNet. The terms from different schemas having similarities greater or equal to 0.8 are considered similar. Bylygin et al. [12] devised an ontology and schema matching approach by combining lexical and semantic similarity with machine learning approaches. The authors used lexical and semantic measures as features and trained various machine learning algorithms including Naive Bayes, logistic regression, and gradient boosted tree. The result achieved showed that the combination of algorithms outperformed the single modal.

Martono et al. [13] provided overview of previous studies related to linguistic approaches used for schema matching. Linguistic methods focused on finding strings and evaluate there similarity in different schemas. The string are normally normalized before to align both the strings before similarity comparison. The normalized strings are categories based on the information relatedness and element with similar category are compared using various similarity measure including Jaro-distance, Lavenstein (edit-distance), and many more. Alwan et al. [14] have summarized the techniques used in the literature for schemas and instances based schema matching. The information used for schema matching is categories into schema information, instance and auxiliary information. Most of the searchers have used syntactic techniques (including n-gram, and regular expression), semantic techniques (including Latent Semantic Analysis, WordNet/Thesaurus and Google Similarity) for schema level and instance level matching to achieve the final goal of data/information interoperability. Kersloot et al. [15] performed a comprehensive systematic review to evaluate natural language processing (NLP) algorithms used for clinical text mapping onto ontological concepts. The findings of the studies were evaluated with respect to five categories; use of NLP algorithms, data used, validation and evaluation performed, result presentation, and generalization of results. The authors revealed that over one-fourth of the NLP algorithms used were not evaluated and have no validation. The systems that claimed generalization, were self evaluated and having no external validation.

Xu et al. [16] presented a framework for discovering indirect links besides direct links among schema elements. The indirect matches were detected for relations such as union, composition, decomposition, selection, and boolean. The indirect links are useful to handle concepts merge, split, generalization, and specialization.

The matching techniques utilized in the study considered terminological relationships (word synonym and hypernym), structural characteristics, data-value characteristics, and expected data values. The experimental results revealed framework effectiveness by achieving more than 90% precision and recall for direct and indirect link matching.

A comprehensive survey from 176 experts including physicians and nurses was conducted by Moll et al. [17] to check their perspectives regarding patient accessible electronic health record (PAEHR). The authors discovered that the PAEHR positively effect after six years of operations despite negative expectations. The primary concerns revealed was logger meeting time, change in documentation practices, and increasing varies of patients regarding their health conditions. However, attitude of both healthcare service providers and patients are changing positively with respect to PAEHR and its benefits.

Methodology

Healthcare interoperability, with a focus on non-standard compliant medical schema, is dependent on the generation and validation of schema maps, as discussed above. To this end, the creation of a cohesive workflow is of utmost importance. In our earlier work [8] we used maximum sequence identification and matching using suffix trees for syntactic matching of two distinctly sources data schemas. This was followed by semantic concept enrichment and subsequently concept matching, for creating rules in the form of schema maps. The simplified mapping functions, thereby created, provided a simple methodology for converting semi-structured medical data, into a non-persisted, interpretable, model form. **In our current methodology we have utilized state-of-the-art machine learning techniques for converting medical schemas into semi-structured form, which is then used to create embedding vectors. These vectors are then used for** modeling and creating interpretable rules for applying semantic reconciliation in the form of schema matching and/or transformation. These rules are managed by the RDR which allows fast inference using the relevant knowledge sub-tree, and can well manage knowledge evolution to scale out (by incorporating new schema) and up (by incorporating changes and additions to existing schema) In essence, the aim here is to define a uniqueness property ($\mathbb{P}$), as shown in Equation (1), whereby the semantic and syntactic uniqueness of each attribute as a disjoint union, is used.

$$range(\mathbb{P}) = \tau_{syntatic} \uplus \tau_{semantic} \tag{1}$$

The syntactic uniqueness as represented in Equation (2), is also a disjoint union of the measured attribute type and its valid values. The type of the attribute can correspond to one of Long (signed decimal values), Double (signed floating point values), Date (yyyy-MM-dd), DateTime (yyyy-MM-dd'T'HH:mm:ss. SSSXXX) or String. On the other hand, "validValues" are determined based on what is contained in some sampled portion of the database. It is represented by the set of all possible categorical values (such as values of the attribute with type Date, DateTime, and String which can be used to represent names, address, and others textual elements of the EMR), the minimum and maximum values enclosed within closed intervals ("[ ]") to represent a range of numerical values (such as values of attributes with type Double or Long),

The elements of t set have been determined based on some common types available to our java application. This set can be further extended to include other types, such as "Boolean", "Long", "UUID" and so on, however each type requires a parser, which can identify the type, based on the values in the instances. As an example, if we add the type "Boolean" to the set t, then a parser must exist, which can at the very least validate, values such as "True", "T", "1", as true, and "False:, "f", "0" as false. However, the problem here lies in determining, if the string "1" and "0" really belongs to the type Boolean or Long? Therefore, in our current implementation, we have limited the attribute type set, as mentioned above. Additionally, while database schema used by EMR systems, correspond to the SQL datatypes[18], the application of semantic reconciliation-on-read strategy, instead necessitates the usage of common programming primitive types. This ensures that the querying application is able to correctly transform the textual values into their most relevant, programming type counterparts.

$$\tau_{syntatic} = \{t \mid t\epsilon\,\{\{Long, Double, Date, DateTime, String\} \uplus validValues\}\} \quad (2)$$

The semantic type is represented by the Equation (3), which is the set of all concepts corresponding to the leaf nodes of the suffix tree, generated from the name of the attribute. Suffix trees are used to divide a string into components, which is a very useful base for quickly identifying the longest common subsequence between a pair of strings and data compression. In our case, these trees provide a manageable list of words, contained with the name of attribute. For each substring, represented by a path of the tree, we then identify the corresponding concept, if it exists, thereby producing a set of concepts which may represent the attribute.

$$\tau_{semantic} = \{s \mid s\epsilon\,\{concept_{i,j} \mid i \leq width(suffixTree(s)) \wedge j\epsilon\,\{substring_i\}\}\} \quad (3)$$

Finally, the set AA of Amplified Attribute (AA), holds all unique attributes of participating schema, as shown in Equation (4). For each element of this set, the attribute is unique if its properties from Equation (1), are not similar to any other element of this set. The similarity function "$\sim$" is a threshold based loose bound, on the disjoint elements of "$\mathbb{P}$".

$$\exists a\epsilon AA \mid (\mathbb{P}(a) \wedge \forall b\epsilon AA \mid \mathbb{P}(b) \rightarrow a \sim b) \quad (4)$$

As a possible realization of this theoretical set notation form, and from a practical point of view, our methodology is presented in the following subsections.

Schema acquisition

In the first step of our semantic reconciliation methodology, we simulate medical data acquisition from five distinct EMR storage systems (S). These include patient reports from OpenEMR (s_1), 100,000 patient records from EMRBOTS (s_2) [19], custom database design by Pan et. al(s_3) for supporting regional clinics and health care centers in China [20], clinical knowledge discovery tool MedTAKMI-CDI (s_4) [21], and our custom implementation (s_5). Each of these medical systems follow the relational database design, with logical entities, such as demographics, diagnosis, medicine or others, placed into tables which can be further linked to one or more tables. While the database design implemented by each of these systems, fulfills the need of their respective information processing applications, the lack of interoperability, in terms of identifying similar attributes or exchanging the medical data is very much evident here. A similar notion of data heterogeneity, in terms of medical data schema, is evident across the healthcare domain. This is caused by various factors, including the lack of one all-encompassing, and universally applicable terminological standard and different normalization level for representing attributes.

In the former case, while SNOMED-CT provides a mechanism for identifying the standard codes for clinical terms and LOINC can be used for laboratory related terms, most attribute names are created based on the gut feeling of the database designer. Additionally, while these codes can be used to represent data instances, the data schema, achieves no benefit from the same. Consider the terms “name” and “patientName”, which refer to the same attribute of the patient entity. However, since there is no standard way to represent this attribute, both are considered correct (s_1 and s_3 use the former representation, while s_2 and s_5 use the latter).

In the later case, differences in normalization cause semantic differences, due to which some data could be available in one schema but absent in others, such as OpenEMR demographics identifying the patient’s residential location using specific attributes like “Address”, “City”, “State”, “Postal Code”, “Country”, and others. Similarly, “EncounterDate” from s_5 is semantically similar to “BeginDate” of “openemr_MedicalProblems” table in s_1, “AdmissionStartDate” of “LabsCorePopulatedTable” in s_2, “time” in “Diagnosis” table of s_3, and “dateOfAdmission” in “Diagnosis” and “CareHistory” tables of s_4. Finally, s_1 and s_3 have separate tables containing the medicinal prescription details, however the same details are unavailable in s_2, s_4, s_5. Once again, this is not an incorrect behavior since this information, might not be a part of the context or the requirements for these EMR storage systems. In fact, the change in context of the EMR storage system from the initial time of development to a later stage of collaborative processing systems, is the main cause of heterogeneity. In order to provide an interoperable solution, it is therefore necessary to provide syntactic and semantic mappings, while taking the data instances (or tuples) into account. Specifically, the creation of Amplified Attributes (AA), as an enriched version of EMR attributes, provides a semantic context, by establishing its existential context(such as schema, table, version, date and other factors), probable data type, possible value set, and its related semantic concepts.

From Attribute to Amplified Attribute

metadata In order to process the EMR schema set S and produce AA, we use the methodology shown in **figAlgorithm**. First we use the data representation s_i, generated through the process explained above () to collect the various medical fragments in memory. We then iterate over these fragments, building a set of attributes (not AA), distinguished by their name, schema's name, table's name, schema's version, source, and recorded data. This entails that "PatientID" from each of the four tables in s_2, and "patientID" from five tables in s_4, would result into nine attributes (assuming, as in the current case, of no differences in versions of these systems). For each attribute, we then generate the suffix array, which provides all possible substring representations contained within the attribute name. In order to generate the set of suffixes, we employ three strategies, forward suffix generation, whereby for a word w of length n, $n-1$ suffixes of size 2 to $n-1$ are produced, backward suffix generation, to produce $n-1$ suffixes in reverse order with size $n-1$ to 2, and regular expression based suffix generation, which splits each word on, change of case, special characters(such as -, _, !, and others), and numbers. An example of this suffix generation process is shown in **figSuffixAttribute**. Here, the attribute "DOB" in s_1, "PatientDateOfBirth" appears in s_2, "birthday" appears in s_3, "DateOfBirth" in s_5, and "birthDate" in s_4. These attributes refer to the same attribute of the patient entity, however, with the suffix generation process, our aim is to identify all words within these strings. While the forward and backward generation process, generates all possible sequences of characters as suffix, the regex based method, only identifies the changes in the string. In this way a large list of suffixes is generated, which is combined using a "TreeSet" data structure of Java, which internally sorts this list as well.

Similarly the attribute, "dateOfAdmission" appears in s_4, while "AdmissionStartDate". "diseaseNameOnAdmission", and "AdmissionEndDate" appear in s_2. As evident within these strings, while they refer to semantically different entity attributes, syntactically they contain similar elements. The suffixes generated from these strings are shown in **figSuffixAttribute (b)**. The suffix generation process, identified thus far, is only able to generate syntactic suffixes, producing many incoherent and unrelated suffixes. In order to counter this problem, and to limit the list of suffixes within the domain, we then query UMLS, with exact search strategy, looking for the existence of any concepts, against each suffix. In case, no semantic concept is found for a particular suffix, it is removed from the final Suffix Array. On the other hand, if atleast one semantic concepts is found against the queried suffix, it is retained. The list of concepts thus found, are also added to Concept Array of the AA. We continue this process, until all substrings have been processed. This is followed by syntactic type check of the attribute, which is based on its data. Here we determine, if the attribute values correspond to one of the types from t (Equation (2)). Additionally, we collect the values of each corresponding data cell, and determine the list of possible values in each case.

Meanwhile the process continues for the next attribute, then the next table, and finally the next system, till no further processing is possible. The flat enriched schema, following the design shown in **figEnrichedSchema** provides the input to our knowledge base. The logically amplified structure of the attribute, AA, is represented in **figAttributeRepresentation**. Here, the elements of AA are categorized into three parts. “AttributeContext” contains the metadata conforming to an instance of the attribute’s existence, with its name, container table name and schema name, acting as a pointer, and schema version, source, and recorded date providing the version control features. In this manner, the attribute’s fully qualified reference, along with its existential context is identified. “AttributeType” then encapsulates the data type features of this attribute’s instances, with its primitive data type, and values providing a representation of the attribute’s instance. Finally, “Attribute Semantics” holds, the semantic information of this attribute, in the form of its suffix array, CA and Embedding Vector. A pair of AA’s with distinct references, are then used as an input to the schema map generation process, which is explained in the next step.

Schema Map generation

Schema Maps, provide an interoperable bridge between two medical systems $(s_i \wedge s_j)$, by identifying the links between their participating attributes. This identification is based on the schema matching process, shown in Algorithm 1, which operates on a pair of AA, $(aa_i, aa_j) \mid aa_i \epsilon s_i \wedge aa_j \epsilon s_j$, and calculates the similarity score S.

Algorithm 1 Attributes similarity identifier

```
                    Input: AmplifiedAttributes $aa_i, aa_j$
                    Output: Similarity $S$
Similarity $S = 0$;
if $aa_i.AttributeContext == aa_j.AttributeContext$ then:
    $S = 1$;
else
    Syntactic Similarity $SynSim = 0$
    $at_i = aa_i.AttributeType$
    $at_j = aa_j.AttributeType$
    if $(at_i.DataType == at_j.DataType)||(at_i.DataType \Leftrightarrow at_j.DataType)$ then:
        $SynSim = 1$
    end if
    Semantic Similarity $SemSim = 0$
    $a\vec{e}_i = aa_i.AttributeSemantics.TreeEmbedding$
    $a\vec{e}_j = aa_j.AttributeSemantics.TreeEmbedding$
    $SemSim = \frac{a\vec{e}_i \cdot a\vec{e}_j}{\|a\vec{e}_i\|\|a\vec{e}_j\|}$
    $S = (0.5 * SynSim) + (0.5 * SemSim)$
end if
return $S$
```

This algorithmic process starts by comparing the metadata context of the two amplified attributes. The schema name, table name, attribute name, and version are used to establish the context of each attribute, which are then evaluated based on naive string matching of corresponding elements. If the pair refer to the same instance of the amplified attribute the process simply returns 1 as the similarity score. If however, the amplified attribute refer to separate instances, we then apply syntactic and semantic similarity on the pair. Firstly we compare the datatypes of the pair, to determine if they either have the same datatype or the datatypes are convertible. In our current approach *Double* is convertible into *Long* and vice versa, and *Date* is convertible into *DateTime* datatypes and vice versa. However, this step is very much implementation specific and can be extended if t is enhanced with newer data types. This test is used to set the “SynSim” score to “1”, if the datatypes are equal or convertible, and “0” otherwise. Secondly, we compare the semantic concepts of the two amplified attributes, by applying **cosine** similarity between their embedding vectors. In order to generate the embedding vectors, we utilize BERT and apply it on the sentence formed by flattening the AA. This operation is applied on each AA in the schema map pair, by concatenating the name of the attribute, its m suffixes, and their associated $0..n$ concepts, to create a sentence as shown in Equation 5:

$$\{Attribute\_Name < w_0, c_{00}, c_{01}, ...c_{0n}; w_1, c_{10}, c_{11}, ...c_{1n}; ...w_m, c_{m0}, c_{m1}, ...c_{mn}; >\} \\ | w_i \epsilon \phi_{suffix}(Attribute\_Name) \wedge c_{ij} \epsilon \phi_{concept}(w_i) \tag{5}$$

An example of the resultant sentence is shown as follows:

“$DateOfAdmission < mission, C0026219; Dat, C1420213, C3813713, C0114838, C0200555, C1439340, C2826247, C0057150, C0002395; Of, C1705644, C0919490, C1704913, C1426958, C0332285, C0456628, C0029151, C1879775; ion, C1700702, C0022023; Admission, C0809949, C0184666; Da, C3668815, C0678223, C0226032, C1602374, C0280573, C4761311, C0332151, C1704643; Date, C1548309, C2740799, C0011008, C1547350, C2348077; on, C2985244, C0029446, C1420342, C1720294, C1720176 >$”

Since the embedding vector is based on the amalgamation of the suffix strings and their corresponding concepts, **cosine similarity can give a good measure of the direction of these vectors**. “SemSim”, thereby obtained is then used in conjuction with previously obtained “SynSim” to calculate the similarity of the two amplified attributes. Using equal weights for syntactic and semantic similarity, we then re-scale their individual values to finally provide a similarity score between “0” and “1”.

The schema matching approach represented above, can be further improved by enwrapping the individual calls with memoization, which can avoid redundant checks. However, in practical terms, the creation of the schema map between two amplified attributes is rarer than its application to transform or link two distinct schemas. Additionally, just like its other schema/ontology matching counterparts, the presented Algorithm 1, applies a one-way compression on the various matched factors to produce a similarity measure/ indicator. This operation is very useful for building unsupervised and semi-supervised automated systems, which base their decisions on a measured confidence score. However, in the healthcare domain, knowledge acquisition and application process, requires the use of supervised automated systems, which can provide strong traceability for the decision making process. Summarily, it is not enough for our schema matching process to simply return S.

This is where, Ripple Down Rules (RDR) knowledge base, steps in. RDR, by design, provides a safe and traceable data structure for creating, maintaining, and evolving knowledge, in the form of iterpretable rules [22, 23, 24]. As **figRDR** shows, the RDR implementation, which serves as our schema map knowledge base, is very closely related to our schema matching algorithm. Beyond the root node of the

RDR, which represents the default condition, at lower depths, syntactic matching characteristics are represented, while semantic matching characteristics, such as the concepts associated with semantic types, follow thereafter. “AttributeType” elements representing the data types of the attributes (from set t) are placed in the first case, followed by convertible types, such as the case where the attribute values, in the form of Long are placed in a string (“*1*”,“*2*”, and so on). The “Except” clauses then point towards various “**SemanticType**”, collected from UMLS. Here, the conditional part of the node refers to types such as (“*Sign or Symptom*”, “*Organism Function*”, “*organism Attribute*”, and so on), while the conclusion part refers to concepts, such as (“*Fever*”, “*blood pressure*”, “*age*”, and so on). In this way, the RDR based knowledge base, provides us with a concrete, memoized, tree data structure, that can be used for schema matching, on the fly. In moving from root to the leaf nodes of this tree, we gain additional matching characteristics, which are all accumulated in the returned results to provide traceability for the decision makers. Additionally, the RDR creation and subsequently evolution process, is dependent on approval by the expert, in determining which cases can be added to the knowledge base. This combination of automated process for building the schema maps, followed by expert intervention to approve the results, enables a safe and consistent knowledge base.

Schema Map application

Experimental Setup

In our earlier work [8] s_1, s_2, and s_5 were used to generate over 115 million patient records, which are converted into a semi-structured form and stored in Hadoop Distributed File System (HDFS). We extended the same setup to create an additional 100,000 records, for 1000 patients with 3 medical fragments for s_1, s_2, and s_4, and 97 randomly selected and generated medical fragments amongst s_1, s_2, s_3, s_4 and s_5. These fragments, follow various design elements, supporting a variety of valid relational storage architectures. Such as, s_1, s_2 and s_4 are represented by creating a separate medical fragment for each participating table, s_3 utilizes its medical fragment to generate a linked record (from a linked object graph), where by the attributes can refer to other objects besides the elements of t, mimicking the application of explicit foreign keys, and s_5 is a flat table structure. The code to generate this data set is available at “uhp_map_generation”[1]. This application produces three custom formatted files, containing an index for patients, an index for their medical fragments, and the medical fragment, corresponding to the EMR data. Using the medical fragments file, we then generate the AA[2]. The resulting set of AA are temporarily stored in a “json” file, which is then read by the same application to partially generate the schema maps. This process, is used to create 21,809 distinct pairs of attributes, across s. Each pair also contains the “relationshipList”, which stores either the positive result of a case insensitive match between the attributes, in which case no further processing is performed, or the results of fuzzy string matching[25] [3] between the attribute names. The “json” file thus produced,

[1] https://github.com/desertzebra/UHP_v4/tree/main/uhpr_storage
[2] https://github.com/desertzebra/UHP_v4/tree/main/uhp_map_generation
[3] Java Library: https://github.com/xdrop/fuzzywuzzy

is then used by a python script to perform the syntactic and semantic matching on the attributes. In this script we used the pre-trained Bert base model with nli mean tokens[26] to create the sentence embeddings, which are then compared using cosine similarity.

The rationale behind switching the applications at various stages, is to cache the results and create checkpoints for restarting any failed stages, easily. Additionally, since python provides better support for easy generation of embedding vectors, it was thus preferred over the Java based implementation, which is otherwise very beneficial for other tools. These applications were executed on a workstation running Ubuntu 20.04.2 LTS on top of AMD Ryzen 3 2200G, and 32GB ram.

In order to compare our computed models with ground truth, **4** data annotators were hired, to anonymously, score the similarity of each pair of attribute names. In order to support this process, we first repurposed one of our generated data matrix, by marking all attribute pairs belonging to the same schema with the symbol “-”. Following this, the annotators, marked each cell corresponding to a pair of attributes, by determining the similarity in terms of dissimilar as “0”, exactly similar as “1”, row attribute as child of column attribute as “¡”, row attribute as a parent of the column attribute as “¿”, and finally, unknown as “ ”. The data sheets generated after this extensive human effort, have been made available for other researchers[4]. These sheets, additionally contain some missing values, which were left out by the annotators, but in order to maintain their originality, these values were not filled; instead during our evaluation for these datasets, the missing values were considered as having the score “0”. Using **Cohen’s Kappa score (d)**, we evaluated the inter-rater agreement of these annotations, which have been visualized in **figKappaScore**. The 7 permutations, amongst the 4 annotators, generate 257 kappa scores, where each individual score corresponds to the correlation of agreement between two annotators. Further, each score identifies the agreement in terms of scoring a set of paired attributes, while keeping the row attribute constant, and moving along the 257 column attributes.

An average dataset was then produced, containing average scores of all annotators, for each attribute pair. This dataset was then used for evaluating the performance of the automated approaches.

Results

The validity of our proposed approach has been evaluated using several techniques, including comparison of the proposed syntactic and semantic matching process with fuzzy string matching, only semantic matching of attribute names using sentence embedding, and with multiple human annotated versions of the schema map. Using the 21,809 AA pairs, we created a two dimensional matrix, of AA identifiers, whereby each cell refers to a “SchemaName_TableName_AttributeName” element. All nodes, for which the SchemaName part of the name is same, are then marked with “-”, and these are omitted from further processing. This various versions of this dataset (D) are available at **url**.

The results of fuzzy string comparison (D_f) on the attribute names of each pair, can be visualized using the heatmap shown in **figFuzzyWuzzy**. Similarly, the results of comparison using BERT based word embedding of the attribute names (D_b),

[4] https://github.com/desertzebra/EMR-Interoperability/tree/master/Implemenation/Data/Annotated

is shown in **figBase**. Finally, the heatmap of the syntactic and semantic matching of the AA's using our proposed Algorithm 1 (D_p), is shown in **figSynAndSemMatching**. The values, for each cell range between "0" and "1", with higher values indicating a larger match between the elements, in each of these cases. The black areas of these heatmaps, indicate missing values, while white indicates a matching value of "0". The few areas with red color, indicate the values above "0.9", while blue areas indicate values above "0.8".

Overall, the number of similarities above the threshold values of "0.8" are greatly increased from D_f, to D_b, and then to D_p (proposed methodology). Changing the threshold value in D_p to "0.9", brings it relatively closer to D_b. This is due to our proposed similarity matching methodology, which gives an equal score (0.5) to the values obtained from both syntactic and semantic matches. Since the decision made at the syntactic evaluation is binary (0 or 1), thus for a similarity score of "0.8", in D_p, "0.5" represents the syntactic similarity, and "0.3" semantic similarity. The semantic similarity value, obtained from this rescaling process, would in turn represent an original value of "0.6" from D_b. Therefore, for an equal semantic representation, the value of semantic similarity after rescaling (between 0 and 0.5) should be "0.4", and originally "0.8". **figSynAndSemMatching0.9** shows the heatmap, of similarity matrix, obtained from D_p with the threshold values as "0.9".

Generally, a comparison of these heatmaps, indicate an increase in the recall rate of the methodology. This is due to the fact, that our proposed methodology is able to match a larger number of AAs, providing 50% of the score to the syntactic match, thereby increasing the count of cells without a missing value. This approach is also able to reduce the impact of semantic matching from names, which might contain repeated words, such as "patient" but in reality refer to disparate concepts, such as "PatientName" and "PatientDateOfBirth". Another important aspect of our matching process is the inclusion of semantic concepts and suffixes, as a sentence before its conversion to an embedded vector. As discussed earlier, this process is used to reduce the impact of syntactic matches alone, which would give a higher score to the attributes "PatientName" and "PatientGender", which refer to different concepts. The inclusion of concepts behind "Name" and "Gender" part of these names, allow for an improved comparison.

In order to evaluate the accuracy and precision rates of these methodologies, we then compared the results of our computed methods D_f, D_b, and D_p, with the average scored datasets by the human annotators. A visualization of the resultant Cohen's Kappa score for the 257 participating attributes, is shown in **figComputedComparisonKappaScore**. The bar chart shown in this visualization, indicates, an overall greater positive agreement between our proposed methodolody D_p, and the average scores of the annotators. The same is shown by the **orange** colored peaks in this dataset, which in all cases is either equivalent to the other approaches (D_f and D_b), or better than them.

with the ground truth. **annotation comparison**

Acknowledgements
Text for this section...

Funding
Text for this section...

Abbreviations

Text for this section. . .

Availability of data and materials

Text for this section. . .

Ethics approval and consent to participate

Text for this section. . .

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Text for this section. . .

Authors' contributions

Text for this section . . .

Authors' information

Text for this section. . .

Author details

[1]Department of Science, University of Cambridge, London, UK. [2]Institute of Biology, National University of Sciences, Kiel, Germany.

References

1. Geraci, A., Katki, F., McMonegal, L., Meyer, B., Lane, J., Wilson, P., Radatz, J., Yee, M., Porteous, H., Springsteel, F.: IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries. IEEE Press, ??? (1991)
2. Healthcare Information and Management Systems Society: Definition of Interoperability (2013). http://www.himss.org/library/interoperability-standards/what-is
3. Khan, W.A., Hussain, M., Latif, K., Afzal, M., Ahmad, F., Lee, S.: Process interoperability in healthcare systems with dynamic semantic web services. Computing **95**(9), 837–862 (2013)
4. Ali, T., Hussain, M., Khan, W.A., Afzal, M., Hussain, J., Ali, R., Hassan, W., Jamshed, A., Kang, B.H., Lee, S.: Multi-model-based interactive authoring environment for creating shareable medical knowledge. Computer Methods and Programs in Biomedicine **150**, 41–72 (2017)
5. SNOMED Clinical Terminologies. http://www.snomed.org/snomed-ct/five-step-briefing Accessed 2020-03-19
6. LOINC. https://loinc.org/ Accessed 2019-03-19
7. CIMI: Clinical Information Modeling Initiative (CIMI) (2015). http://www.opencimi.org/ Accessed 2020-03-19
8. Satti, F.A., Ali, T., Hussain, J., Khan, W.A., Khattak, A.M., Lee, S.: Ubiquitous Health Profile (UHPr): a big data curation platform for supporting health data interoperability. Computing (2020). doi:10.1007/s00607-020-00837-2
9. Althubaiti, S., Kafkas, Ş., Abdelhakim, M., Hoehndorf, R.: Combining lexical and context features for automatic ontology extension. Journal of biomedical semantics **11**(1), 1–13 (2020)
10. Nozaki, K., Hochin, T., Nomiya, H.: Semantic schema matching for string attribute with word vectors. In: 2019 6th International Conference on Computational Science/Intelligence and Applied Informatics (CSII), pp. 25–30 (2019). IEEE
11. Yousfi, A., El Yazidi, M.H., Zellou, A.: xmatcher: Matching extensible markup language schemas using semantic-based techniques. International Journal of Advanced Computer Science and Applications **11**(8), 655–665 (2020)
12. Bulygin, L.: Combining lexical and semantic similarity measures with machine learning approach for ontology and schema matching problem. In: Proceedings of the XX International Conference "Data Analytics and Management in Data Intensive Domains"(DAMDID/RCDL'2018), pp. 245–249 (2018)
13. Martono, G.H., Azhari, S.: Review implementation of linguistic approach in schema matching. International Journal of Advances in Intelligent Informatics **3**(1), 1–9 (2017)
14. Alwan, A.A., Nordin, A., Alzeber, M., Abualkishik, A.Z.: A survey of schema matching research using database schemas and instances. International Journal of Advanced Computer Science and Applications **8**(10) (2017)
15. Kersloot, M.G., van Putten, F.J., Abu-Hanna, A., Cornet, R., Arts, D.L.: Natural language processing algorithms for mapping clinical text fragments onto ontology concepts: a systematic review and recommendations for future studies. Journal of biomedical semantics **11**(1), 1–21 (2020)
16. Xu, L., Embley, D.W.: Discovering direct and indirect matches for schema elements. In: Eighth International Conference on Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings., pp. 39–46 (2003). IEEE
17. Moll, J., Cajander, Å.: Oncology health-care professionals' perceived effects of patient accessible electronic health records 6 years after launch: a survey study at a major university hospital in sweden. Health informatics journal **26**(2), 1392–1403 (2020)
18. SQL Data Types. https://www.w3resource.com/sql/data-type.php. Accessed: 2021-01-14
19. Kartoun, U.: A methodology to generate virtual patient repositories. arXiv preprint arXiv:1608.00570 (2016)
20. Pan, L., Fu, X., Cai, F., Meng, Y., Zhang, C.: Design a novel electronic medical record system for regional clinics and health centers in china. In: 2016 2nd IEEE International Conference on Computer and Communications (ICCC), pp. 38–41 (2016). IEEE
21. Inokuchi, A., Takeda, K., Inaoka, N., Wakao, F.: Medtakmi-cdi: interactive knowledge discovery for clinical decision intelligence. IBM Systems Journal **46**(1), 115–133 (2007)

22. Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Preston, P., Srinivasan, A.: Ripple down rules: Turning knowledge acquisition into knowledge maintenance. Artificial Intelligence in Medicine **4**(6), 463–475 (1992)
23. Richards, D., et al.: Two decades of ripple down rules research (2009)
24. Kim, D., Han, S.C., Lin, Y., Kang, B.H., Lee, S.: Rdr-based knowledge based system to the failure detection in industrial cyber physical systems. Knowledge-Based Systems **150**, 1–13 (2018)
25. Adam Cohen: FuzzyWuzzy: Fuzzy String Matching in Python (Jul 8th, 2011). https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/
26. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, ??? (2019). http://arxiv.org/abs/1908.10084

Figures

Figure 1 Schemas used for knowledge interoperablity.

Figure 2 Sample figure title

Figure 3 Sample figure title

Tables

Table 1 Sample table title. This is where the description of the table should go

	B1	B2	B3
A1	0.1	0.2	0.3
A2	...	..	.
A3	..	.	.

Additional Files

Additional file 1 — Sample additional file title
Additional file descriptions text (including details of how to view the file, if it is in a non-standard format or the file extension). This might refer to a multi-page table or a figure.

Additional file 2 — Sample additional file title
Additional file descriptions text.