Semantic Bridge for Resolving Healthcare Data Interoperability

Fahad Ahmed Satti
*Ubiquitous Computing Lab,
Department of Computer Science &
Engineering, Kyung Hee University*
South Korea
fahad.satti@oslab.khu.ac.kr

Wajahat Ali Khan
*Ubiquitous Computing Lab,
Department of Computer Science &
Engineering, Kyung Hee University*
South Korea
wajahat.alikhan@oslab.khu.ac.kr

Taqdir Ali
*Ubiquitous Computing Lab,
Department of Computer Science &
Engineering, Kyung Hee University*
South Korea
taqdir.ali@oslab.khu.ac.kr

Jamil Hussain
*Ubiquitous Computing Lab,
Department of Computer Science &
Engineering, Kyung Hee University*
South Korea
jamil@oslab.khu.ac.kr

Hyeong Won Yu
*Department of Surgery,
Seoul National University
Bundang Hospital*
South Korea
hyeongwonyu@gmail.com

Seoungae Kim
*Ubiquitous Computing Lab,
Department of Computer Science &
Engineering, Kyung Hee University*
South Korea
seoungae@oslab.khu.ac.kr

Sungyoung Lee
*Ubiquitous Computing Lab,
Department of Computer Science &
Engineering, Kyung Hee University*
South Korea
sylee@oslab.khu.ac.kr

Abstract—**Data Interoperability is a critical part of achieving healthcare interoperability. In this paper, we present a novel methodology for automating the schema matching process at attribute level, among three non-standard and serialized schemas. We achieved 71.8% mappings, using a four stage process with string matching, longest common substring matching using suffix tree, and ConceptNet lookup.**

Index Terms—**Healthcare, Interoperability, Schema Matching**

I. Introduction

In the last decade, digitization efforts in healthcare service delivery have led to the generation of a very large volume of sensitive and heterogeneous data. This heterogeneity is a consequence of the plethora of available information systems and standards for messaging, terminologies, decision support and others [2]. This has led to a lack of interoperability and co-ordination between medical service providers and consumers [1], eventually leading to risks such as erroneous diagnostics, greater operating cost [3], or non-adherence top treatment plans by the patients [4]. The problem is compounded due to the existence of some healthcare information management systems, which do not utilize any formal standard to build their schemas. Leading technologies such as Big Data (integrated storage), Semantic Mapping, and Blockchain(verification) can be entwined together to build a ubiquitous healthcare platform, which can not only resolve the data heterogeneity problem but also provide access to holistic medical data curation services for large healthcare service delivery platforms.

In building such a platform, we utilized three distinct schemas (OpenEMR, EMRBots, and a custom HMIS implementation) to produce over 117 million, synthesized medical records for 390,000 patients. This heterogeneous data is stored in Hadoop based Big Data Storage engine, in semi-structured form. Using Hive, we are able to query and retrieve error-free results, which are syntactically associated with each other using patient, medical system, and record identifiers. This integrated storage, provides the foundation for applying a novel semantic reconciliation-on-read process for resolving data interoperability. The reconciliation process, in turn, is based on the creation and application of a semantic bridge which provides a link amongst the attribtues of any two schemas. This many to many mapping of attributes and schemas, is generated through an offline process. While the application of the same is an online process, which is delayed until the user request, for some subset of the data, is received. In this manner, a semantic bridge can be used to create a temporary view, which maps and/or transforms the medical data from one schema to another, without compromising its integrity or originality.

In this paper, we present the semantic bridge and its current unsupervised, creation strategy, which produces more than 6,651 maps between 144 attributes in 3 schemas. Using only the schema information and a four stage process, based on string matching, longest common subsequence identification using suffix trees, and ConceptNet relatedness check based on Numberbatch embeddings [5].

II. MOTIVATION

A. Healthcare Data Interoperability

Interoperability represents the policies and guidelines, which when implemented, can bridge the gap between systems and services. Data Interoperability is a part of this ecosystem, which focuses on resolving the integration, exchange and consumption of data, while maintaining its context. It requires schema matching techniques that can transform source data into understandable and application ready format [9]. This transformation can be achieved by using a standard based approach or a mediation based one. Here, the former focuses on creating a single standard that contains the common and useful features of various participating standards. While the latter, creates conversion mechanisms to convert standards into a form, acceptable by all participants [10]. Linked Data is one of the well know example of achieving standard based data interoperability, which uses expert intervention to create links within and between a collection of dataset. On the other hand, Semantic Information Layer (SIL) [11] is an example of mediation approach which is based upon automatically linking the existing semantically enriched attributes. Data interoperability approaches can enhance healthcare services quality and reduce its costs by eliminating redundant operations.

B. Data Storage Engines in Healthcare

One of the reasons behind heterogeneity in healthcare data is the utilization of different types of storage engines. Healthcare data storage strategies can be categorized into, Relational data store, Column-Oriented data store, Graph store, Document stores and Key-Value maps.

1) Relational data store: The most common type of storage engine used by medical systems, is the relational database management system. It provides a mechanism for storing data instances/tuples, which are uniquely identifiable with primary keys. Each tuple can also contain foreign keys to identify their relationships, with other instances. This type of data store is easy to use, however, all records must conform to a well-defined schema before insertion. As a result, the retrieval process is relatively fast and guarantees a well-defined result set. The other data stores, are typically called NoSQL and do not require strict schema check on insertion. Instead, they use a schema-on-read strategy for fast insertion and relatively slow retrievals. Formalization of the final result is dependent on the reporting system.

2) Column-Oriented data store: Column-Oriented data stores are very closely related to the relational data stores in terms of their well-defined schema, however, they are optimized for storing, and operating on data columns, rather than rows. This allows, operations on complete columns of data to be optimized for speed. Medical systems relying on column-oriented data stores have shown high scalability and improved performance in comparison with traditional RDBMS, and provide a viable alternative [6].

3) Graph store: Graph stores, for storage and processing of medical data have found some acceptance in research initiatives, such as in [7]. The data generated by these systems is beneficial due to their less stringent schema, more express-ability, easier application of semantics and scalable infrastructure, especially when compared with the relational model.

4) Document stores: Document stores, provide a NoSQL data storage engine which does not require any predifined schema and favors fast insertions and scalability. This is especially suited for storing documents such as free text based documents, images, clinical practice guidelines, and others. The drawback of these storage engines is the relatively slower read operations, requiring complex, deep searches [8] for applying a schema on unstructured documents.

5) Key-Value maps: Finally, the most simplest form of data store is the Key-Value map, which stores completely unstructured data. Typically key based indexes are used to somewhat alleviate the slow nature of retrieve operations. These are not typically used as a storage engine for medical information management systems. Instead, they can be used in conjunction with other NoSQL data stores and/or to manage large streaming data such as from medical IoT devices.

C. Semantic Integration of Relational Databases

Data Integration is a complex operation which requires user intervention at various stages. Automating some of the most common tasks, and presenting the knowledge engineer with empirical backing to similarity matching operations can be very useful. [12] have provided a very good overview of the Semantic Data Integration usecases, needs and methodologies. The authors have presented an overview of what an integrated biological store would look like. While the data processing layers, storage mechanisms and integrating technologies are very different from our research, the overall concept of semantic reconciliation-on-read, is similar. The data integration process presented by the authors relies on SPARQL queries with a pre-defined mapping for underlying datastores. W3C has provided R2RML [13] as a standard for converting relational data (RDB) into resource description format (RDF) model. Detailed review of the tools supporting R2RML and other technologies for mapping relational data to RDF is presented in [14]. As identified by the authors in this study, R2RML is not an all encompassing standard and many of the current implementations are restricted by their support for data sources. Another important factor in these mappings is that in most cases they are dependent on how well the knowledge engineer/programmer has defined the mappings or enriched the database with resource descriptions. In this presented work, we have automated the pre-processing of the schema mappings, thereby producing semantic bridges, which in at-least some cases would only require verification from the knowledge engineer and reduce their workload.

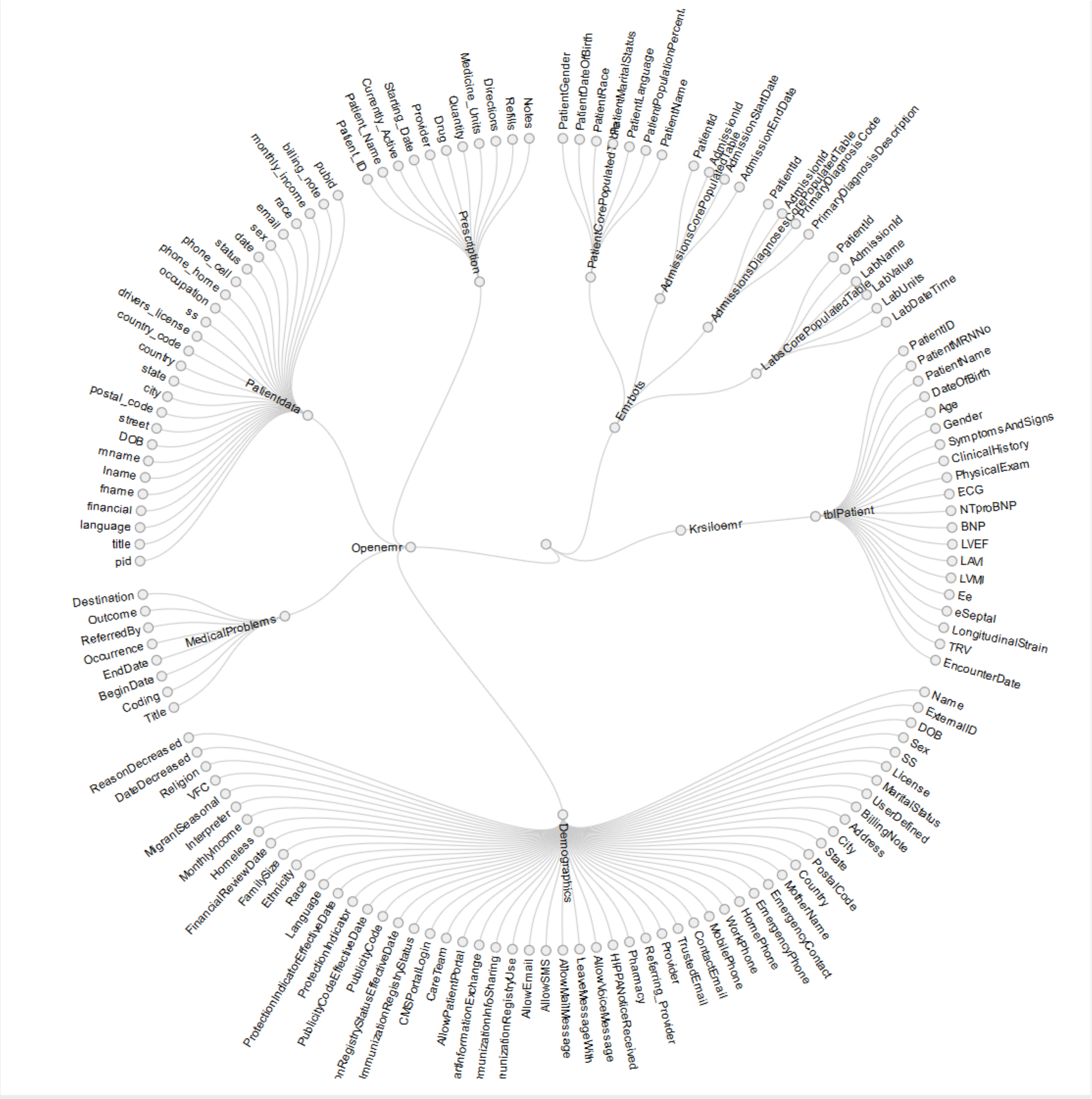
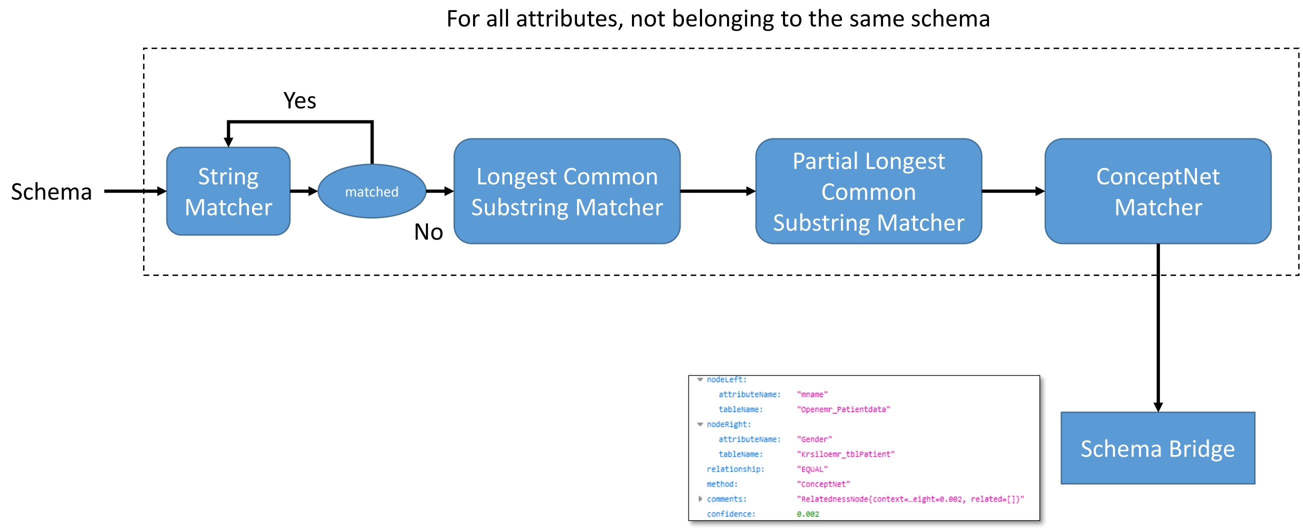


Fig. 1. Circular Dendrogram of participating schemas(generated via https://app.rawgraphs.io/)


For all attributes, not belonging to the same schema
Schema
String Matcher
Yes
matched
No
Longest Common Substring Matcher
Partial Longest Common Substring Matcher
ConceptNet Matcher
Schema Bridge

```
▾ nodeLeft:
    attributeName:   "mname"
    tableName:       "Openemr_Patientdata"
▾ nodeRight:
    attributeName:   "Gender"
    tableName:       "Krsiloemr_tblPatient"
  relationship:      "EQUAL"
  method:            "ConceptNet"
▸ comments:          "RelatednessNode{context=…eight=0.002, related=[]}"
  confidence:        0.002
```

Fig. 2. The proposed Schema Matching process

III. Methodology

The Ubiquitous Healthcare Platform, provides an interoperable, NoSQL based, scalable, data storage platform, which can consume data from various sources and produce a comprehensive medical profile of the patient [15]. For prototype implementation, over 117 million medical records were synthesized. Out of these 8 million records corresponding to OpenEMR and our custom medical silo schema, for 290,000 patients are based on 40 real cardiovascular patients. Another 109 million records were synthesized from the set of 10,000 patient records synthesized by EMRBOT [16]. The 3 schemas (KrSiloEMR, EMRBOTs, and OpenEMR), correspond to 7 medical fragments, and altogether 143 attributes. As can be seen in Fig. 1, the schemas are widely different in terms of their terminologies and brevity. OpenEMR uses the character "_" to split multiple words in some cases and in others it uses word capitalization. OpenEMR contains many attributes containing the patient's demographics details, while EMRBOTs and KrSiloEMR have substantially low details. These and many other problems, necessitate the need to bridge the gaps between the schema and provide a mechanism for linking or transforming one schema into another.

For schema matching we have used a four stage process as shown in Fig. 2. Each stage is applied on a pair of attributes, whereby the left attribute and the right attribute do no belong to the same schema.

The details of these steps are described in the following subsections.

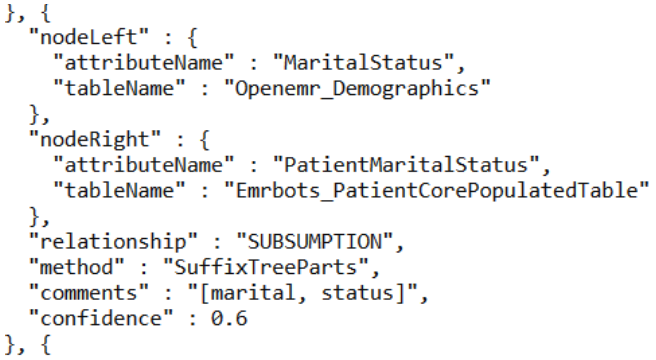
A. String Matcher

In the first stage, we apply a simple, case insensitive, string matcher, for comparing the two attributes. Each attribute is first converted into lowercase, followed by removal of any special characters. If the two attributes match (such as in the case of Krsiloemr.tblpatient.PatientID and EMRBOTS.AdmissionsCorePopulatedTable.patientid) the link between these two attributes is stored and no further processing is perfomed on it. In case the attributes do not completely match, they are processed through the next stages.

B. Longest Common Substring Matcher

In the second stage, we utilize suffix tree method to identify the Longest Common Substring(LCS) between the two participating attributes. For suffix tree we use the Concurrent Tree implementation provided by Google Code, version 2.6.1[1]. Using the default character based factory we build a suffix tree with both attributes. The subsequent tree will contains the common elements of the two attributes. A simple lookup for the longest substring returns the non-leaf node with the longest character sequence. We apply a threshold value of $\frac{2}{3} - Length_{attrLeft}$ to check if this substring qualifies as a subsumption relation or not. The confidence value of this check is calculated using the equation "(1)".

[1] https://mvnrepository.com/artifact/com.googlecode.concurrent-trees/concurrent-trees/2.6.1



```
}, {
  "nodeLeft" : {
    "attributeName" : "MaritalStatus",
    "tableName" : "Openemr_Demographics"
  },
  "nodeRight" : {
    "attributeName" : "PatientMaritalStatus",
    "tableName" : "Emrbots_PatientCorePopulatedTable"
  },
  "relationship" : "SUBSUMPTION",
  "method" : "SuffixTreeParts",
  "comments" : "[marital, status]",
  "confidence" : 0.6
}, {
```

Fig. 3. Partial Longest Common Substring Matching result

$$Confidence = 1 - \frac{Length_{attrLeft} - Length_{LCS}}{Length_{attrLeft}} \tag{1}$$

C. Partial Longest Common Substring Matcher

Due to the nature of database schemas and a high possibility of multiple words existing in each attribute name, it is much more feasible to do a partial LCS of the attributes. This ensures that if we are able to split the right attribute into words, each of those words would be matched with the attribute left. To achieve this, we use an additional loop over the words of attribute right, identified using a regular expression, which splits the attribute string, using case changes, digits, or any special characters in the text. Then for each word, we build a suffix tree with the left attribute. For each word we check if the longest common substring is greater than $\frac{2}{3} - Length_{wordRight}$. A final check then ensures that atleast one word in the right attribute has the previous condition as true. Fig. 3 shows the result of one such matching, whereby the words “Marital” and “Status” from the right attribute “PatientMaritalStatus” were matched with the left attribute “MaritalStatus”.

D. ConceptNet Matcher

A final similarity check between the two attributes is made using a local copy of the ConceptNet API [5] and the “relatedness” query. This query uses numberbatch.h5 embeddings to calculate a score for the relationships between two concepts and returns the top result. Here, both the left attribute and the right attribute were split into words before querying ConceptNet. As a result, for each pair of words in each attribute we were able to compute a relationship. The relationship was marked as “equal”, if the relatedness weight was greater than 0.9, it was marked as ambigious if the relatedness weight was greater than 0, and skipped otherwise. A final entry in the Schema Bridge was added with the average weight of all words and including the individual matching results as well.

IV. Results

Our prototype implementation of the semantic bridge creation process, as presented above, is in java, which took 47 mins and 46 seconds to produce 6,651 mappings for 144 attributes (22 for EMRBOTS, 20 for Krsiloemr, and 102 for OpenEMR), out of a possible 9,204 (2,440 for EMRBOTS, 2,480 for Krsiloemr, 4,284 for OpenEMR). Stage wise matching results are shown in table I. Here, string matching, in stage 1, provides the most accurate result for only 10 attribute pairs. Stage 2 has no results, due to the very strict check of 2/3 similar word length. Stage 3 produces 41 mappings, which get a secondary check in stage 4. The largest number of matchings (6,600) were produced in stage 4, using the related concept match. In this way, we were able to automatically identify 71.8% of the possible mappings, where atleast some relationship exists.

TABLE I
Stagewise schema matching results

Stage	Matched Results
1. String Matcher	10
2. Longest Common Substring Matcher	0
3. Partial Longest Common Substring Matcher	41
4. ConceptNet Matcher	6600

V. Conclusion and Future Work

Semantic Data Integration in the absence of expert intervention is a difficult task. Even with expert intervention, a general approach would prove to be too tedious, especially in the presence of new schema generation and schema evolutions. In this paper, we have proposed a methodology for automating the schema matching task and producing Schema Bridge at attribute level to support the knowledge engineer.

In future, we shall look towards integrating UMLS lookup as an additional stage for producing more domain specific mappings. We shall also integrate this process with our data interoperability platform, and test it on patient data.

Acknowledgment

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-0-01629) supervised by the IITP(Institute for Information & communications Technology Promotion)”and IITP-2017-0-00655 and NRF-2016K1A3A7A03951968 and NRF-2019R1A2C2090504.

References

[1] L. Samal et al., “Care coordination gaps due to lack of interoperability in the United States : a qualitative study and literature review,” BMC Health Serv. Res., pp. 1–8, 2016.

[2] M. L. M. Kiah, A. Haiqi, B. B. Zaidan, and A. A. Zaidan, “Open source EMR software: Profiling, insights and hands-on analysis,” Comput. Methods Programs Biomed., vol. 117, no. 2, pp. 360–382, 2014.

[3] K. M. Mcdonald, C. L. Bryce, and M. L. Graber, “The patient is in : patient involvement strategies for diagnostic error mitigation,” no. August, pp. 1–7, 2013.

[4] K. B. Haskard Zolnierek and M. R. Dimatteo, “Physician communication and patient adherence to treatment: A meta-analysis,” Med. Care, vol. 47, no. 8, pp. 826–834, 2009.

[5] R. Speer, J. Chin, and C. Havasi, “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge,” no. Singh 2002, pp. 4444–4451, 2016.

[6] A. Celesti, M. Fazio, A. Romano, A. Bramanti, P. Bramanti, and M. Villari, “An OAIS-based hospital information system on the cloud: Analysis of a NoSQL column-oriented approach,” IEEE J. Biomed. Heal. Informatics, vol. 22, no. 3, pp. 1–7, 2018.

[7] I. Balaur et al., “EpiGeNet: A Graph Database of Interdependencies Between Genetic and Epigenetic Events in Colorectal Cancer,” J. Comput. Biol., vol. 24, no. 10, pp. 969–980, 2017.

[8] W. L. Schulz, B. G. Nelson, D. K. Felker, T. J. S. Durant, and R. Torres, “Evaluation of relational and NoSQL database architectures to manage genomic annotations,” J. Biomed. Inform., vol. 64, pp. 288–295, 2016.

[9] P. Pagano, L. Candela, and D. Castelli, “Data Interoperability,” Data Sci. J., vol. 12, no. 0, pp. GRDI19–GRDI25, 2013.

[10] S. A. Renner, J. G. Scarano, and A. S. Rosenthal, “Data interoperability: Standardization or Mediation,” 1st IEEE metadata Conf., pp. 1–8, 1996.

[11] F. Song, G. Zacharewicz, and D. Chen, “An ontology-driven framework towards building enterprise semantic information layer,” Adv. Eng. Informatics, vol. 27, no. 1, pp. 38–50, 2013.

[12] A. C. Sima, K. Stockinger, T. M. de Farias, and M. Gil, “Semantic Integration and Enrichment of Heterogeneous Biological Databases,” in Evolutionary Genomics: Statistical and Computational Methods, M. Anisimova, Ed. New York, NY: Springer New York, 2019, pp. 655–690.

[13] “R2RML: RDB to RDF Mapping Language,” 2012. [Online]. Available: https://www.w3.org/TR/r2rml/. [Accessed: 05-Nov-2019].

[14] F. Michel, J. Montagnat, and C. Faron-Zucker, “A survey of RDB to RDF translation approaches and tools,” Informatique, Signaux Et Systèmes, p. 23, 2014.

[15] F. A. Satti, W. A. Khan, G. Lee, A. M. Khattak, and S. Lee, “Resolving Data Interoperability in Ubiquitous Health Profile Using Semi-structured Storage and Processing,” in In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC’19), 2019, pp. 762–770.

[16] U. Kartoun, M. General, and H. Harvard, “A Methodology to Generate Virtual Patient Repositories,” CoRR, vol. abs/1608.0, 2016.