BUILD-KG: Integrating Heterogeneous Data Into Analytics-Enabling Knowledge Graphs

Kara Schatz, Pei-Yu Hou, Alexey V. Gulyuk, Yaroslava G. Yingling, and Rada Chirkova,
North Carolina State University, Raleigh, North Carolina, USA
Emails: kmschat2@ncsu.edu, hpeiyu@ncsu.edu, agulyuk@ncsu.edu, ygyingli@ncsu.edu, rychirko@ncsu.edu

2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386570

***Abstract*—Knowledge graphs *(KGs)*, with their flexible encoding of heterogeneous data, have been increasingly used in a variety of applications. At the same time, domain data are routinely stored in formats such as spreadsheets, text, or figures. Storing such data in KGs can open the door to more complex types of analytics, which might not be supported by the data sources taken in isolation. Giving domain experts the option to use a predefined automated workflow for integrating heterogeneous data from multiple sources into a single unified KG could significantly alleviate their data-integration time and resource burden, while potentially resulting in higher-quality KG data capable of enabling meaningful rule mining and machine learning.**

In this paper we introduce a domain-agnostic workflow called *BUILD-KG* for integrating heterogeneous scientific and experimental data from multiple sources into a single unified KG potentially enabling richer analytics. BUILD-KG is broadly applicable, accepting input data in popular structured and unstructured formats. BUILD-KG is also designed to be carried out with end users as humans-in-the-loop, which makes it domain aware. We present the workflow, report on our experiences with applying it to scientific and experimental data in the materials science domain, and provide suggestions for involving domain scientists in BUILD-KG as humans-in-the-loop.

***Index Terms*—Integrating heterogeneous data into knowledge graphs, domain-agnostic integration workflow enabling richer data analytics, domain experts as humans in the loop.**

I. Introduction

Recent years have seen a rise in the popularity of knowledge graphs *(KGs)* in many applications. KGs store real-world facts in the format of *subject-predicate-object* (s, p, o) *triples*, where the subject s and object o are KG nodes representing real-world entities, and the predicate p indicates the real-world relationship between them. This format can be used to flexibly encode large-scale heterogeneous data, making KGs well suited for a variety of applications. At the same time, domain data are in many cases routinely stored in other formats, e.g., as spreadsheets, text, or figures. While such formats can be familiar and intuitive to users, storing the same data in KGs can open the door to more complex types of analysis, such as rule mining and machine learning. Moreover, combining heterogeneous data from multiple sources into a single unified KG could lead to even richer analytics not supported by the sources taken in isolation. As such, the KG format can be preferable to other data-storage formats in many scenarios.

This research has been supported by the National Science Foundation under Grant No. CBET-2019435.

Consider a **motivating example** arising from the use case that we work with in this paper. Figs. 1(a)-(b) show fragments of the large-scale data coming from two materials-science teams working in the Science and Technologies for Phosphorus Sustainability (STEPS) Center.[1] While both teams study interactions between phosphate-binding proteins *(PBPs)* and phosphate ions in solvents, they do not use the exact same materials, nor do they use the same experimental procedures or settings. Moreover, they do not use the same storage format for their data: The first team stores their data as *spreadsheets* in (Data) Source 1, see Fig. 1(a), while the second team stores data as *regularized text* and *images* in Source 2, see Fig. 1(b).

The research teams would like to *improve the utilization of their large-scale scientific and experimental data*, by *integrating the data into a single unified KG*. Fig. 1(c) shows one such possible KG, which would allow the researchers to accelerate scientific discovery compared to what could be supported by their isolated source data. The desired integration process would involve conversion of the heterogeneous source data into the KG format. It would also involve combining the resulting KG fragments in a way that would ensure overlap in the shared entities, with potential addition of extra connections across the converted sources, see the dashed edges in Fig. 1(c).

Integrating their data into a KG might not be trivial for the research teams to accomplish on their own. Further, adding the extra connections across the converted sources might be a challenge in case the teams are not very familiar with each other's projects. These issues could make it difficult for unassisted domain scientists to integrate their data effectively into the unified KG format that could enable richer analytics.

Given enough time and resources, domain scientists could certainly solve their KG-integration problem in a one-off way for their particular purpose. At the same time, their being able to use instead a predefined automated workflow for integrating heterogeneous data from multiple sources into a single unified KG could significantly alleviate the time and resource burden, while potentially resulting in higher-quality KG data conducive to accelerating scientific discovery. Ideally, such a workflow would allow the scientists to input their data in familiar formats and to control the KG ontology. Furthermore, it would output an integrated KG that would reflect the data-analysis expectations of the scientists, allowing them to make adjustments as needed in the integration process.

[1] https://steps-center.org/

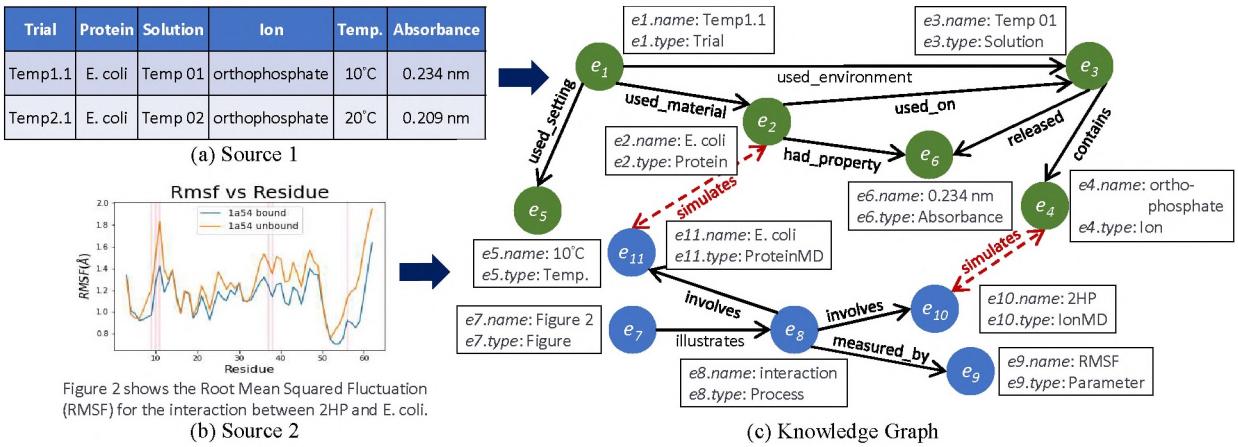

Fig. 1. The motivating example for our proposed BUILD-KG workflow, showcasing a scenario in which data of different types from multiple sources (a)-(b) need to be converted into a single unified KG (c). Source 1 (a) provides *spreadsheet data*, while Source 2 (b) provides *images* and *regularized text* data. A KG capturing and connecting these data is shown in (c). The dashed edges in (c) provide examples of relationships between entities across the data sources.

In this paper we introduce a domain-agnostic workflow called *BUILD-KG* for integrating heterogeneous scientific and experimental data from multiple sources into a single unified KG potentially enabling richer analytics. By design, BUILD-KG is broadly applicable, accepting input data in popular structured and unstructured storage formats. To enable appropriate processing of domain-specific data, it accepts inputs from domain scientists regarding the semantics and handling of the data, in an effort to ensure that the resulting KG will be accurate and useful for their needs. This makes BUILD-KG *domain agnostic* and *domain aware* at the same time. Moreover, the workflow is designed to be carried out with end users as humans-in-the-loop. In this way, BUILD-KG enables domain scientists to facilitate the KG construction and verify that the end result will align with their expectations, potentially enabling acceleration of scientific discovery.

To the best of our knowledge, most KG-construction approaches are not analogous to ours, as their efforts focus only on textual data, see, e.g., [1]–[3], and/or are domain-specific, see, e.g., [4]–[11]. The approach of [12], which converts data in multiple formats to the KG format, is domain independent, but does not directly involve domain experts as humans-in-the-loop. The KG-construction procedure of [13], which also accepts non-textual data inputs from users and is applicable to multiple domains, is complementary to our work, as it only supports data in the JSON and audio formats.

Our **specific contributions** are as follows:

- We propose a domain-agnostic, human-in-the-loop workflow called BUILD-KG to construct KGs from structured and unstructured data according to domain experts' specifications, which makes BUILD-KG domain aware;
- Within BUILD-KG, we introduce a collection of conversion procedures for three popular data types: spreadsheet data, images with annotations, and regularized text data;
- We propose a BUILD-KG methodology for combining multiple heterogeneous data sets into a unified KG;
- We outline our implementation of the BUILD-KG workflow, and report on our experiences with applying it to scientific and experimental STEPS-center data; and
- We report on our experiences working with STEPS researchers, and provide tips on involving domain scientists as humans-in-the-loop in the BUILD-KG workflow.

The remainder of this paper is organized as follows. We review related work in Section II and provide a problem statement in Section III. In Section IV we introduce the proposed BUILD-KG workflow, illustrating it in Section V with a STEPS-center materials-science use case. In Section VI we describe the role of humans-in-the-loop in the BUILD-KG workflow. We conclude in Section VII.

II. Related Work

Our work is most closely related to the topic of knowledge-graph *(KG)* construction, see, e.g., [1]–[3], [12]–[15]. KG construction is a complex process, with approaches ranging from fully manual to semi-automatic to fully automatic. Since fully manual construction is not scalable and fully-automated construction is error prone, most KG-construction approaches, including ours, are semi-automatic.

The most common format of source data in KG construction is text, with many approaches, such as [1]–[3], supporting textual inputs only. These approaches rely on machine learning, most commonly natural-language processing *(NLP)*, to extract data. In contrast, our BUILD-KG workflow is specifically designed to handle a variety of input types, including some of the most popular data-storage formats, so that domain scientists can use BUILD-KG to convert their data to the KG format while continuing their regular data collection. The domain-independent approach of [12], which also converts multiple data formats to the KG format, does not directly involve domain experts, so it is unclear how much control they would have over the handling of their data. Moreover, [12] does not discuss how overlapping portions of disparate data are handled; in contrast, our approach covers this scenario, as these "junction points" can be crucial for domain scientists in

their KG exploration. The KG-construction procedure of [13], which also accepts non-textual data inputs from users and is applicable to multiple domains, is complementary to our work, as it only supports data in the JSON and audio formats.

Domain-specific KG construction has become popular due to the nuances present in domain-specific data [14]. Domain-specific KGs have been generated in many domains, including geosciences [4], education [5], science [6], medical records [7], e-commerce [8], power-grid equipment [9], finances [10], and surveying and remote sensing [11]. In contrast, while our use case is in the materials-science *(MS)* domain, our proposed BUILD-KG workflow is entirely domain agnostic. Moreover, our workflow allows scientists in a variety of domains to input their existing data and easily specify their desired KG ontology, which makes our approach domain aware, while at the same time enabling richer analytics on the output KGs.

To handle textual data, our BUILD-KG workflow includes an NLP component that performs named entity recognition *(NER)* [16]. Despite the recent advances in pretrained NER models, most are trained on common-sense and common-knowledge corpora, see, e.g., WikiBERT [17] and bert-base-NER [18], [19]. Such general models may not always perform well on domain-specific texts. Pretrained language models have also been built in some scientific domains, e.g., [20], [21] in the biomedical domain and [22], [23] in the MS domain. However, in our MS use case, these MS pretrained language models cannot achieve a high prediction accuracy, due to the variety in language used by scientists in different subdomains of MS. Thus, in our proposed BUILD-KG workflow, to complete the NER step we use the NLP tool `flair` [24], as it allows users to easily build their own language model.

III. Problem Statement

We define a *knowledge graph (KG)* $\mathcal{G}$ as a 5-tuple $\mathcal{G} = (\mathcal{E}, \mathcal{T}, \phi, \mathcal{P}, \mathcal{L})$, where $\mathcal{E}$ is the set of *entities*, $\mathcal{T}$ is the set of *entity-types*, $\phi : \mathcal{E} \to \mathcal{T}$ is the *entity-type labeling function*, $\mathcal{P}$ is the set of *predicates*, and $\mathcal{L} \subseteq \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ is the set of *triples*. Each element $(s, p, o) \in \mathcal{L}$ is called a *triple* and represents the real-world fact that the *subject* s has a relationship of type p with the *object* o. For example, the triple (*Solution 01*, `contains`, *orthophosphate*) represents the fact that "the solution with ID 01 contains orthophosphate."

We now introduce the formal **problem statement** for the problem addressed by our BUILD-KG workflow: Given a set $\mathcal{D}$ of heterogeneous data files, integrate the data in $\mathcal{D}$ into a unified KG $\mathcal{G} = (\mathcal{E}, \mathcal{T}, \phi, \mathcal{P}, \mathcal{L})$ that could enable richer analytics not supported by the sources taken in isolation.

To limit the scope of this general problem for this paper, we focus on three distinct data types for the data files in $\mathcal{D}$: (1) *spreadsheet data*, (2) *images with annotations*, and (3) *regularized text data*. These data types have been chosen due to their popularity among domain scientists for storing their data. *Spreadsheet data* are relational-type data stored in data sheets in the spreadsheet format. Data sheets contain data on objects stored as rows; the columns headers indicate the specific components of the data objects. These components have relationships between them that are specified in triple sheets and can be encoded as KG edges. *Images with annotations* consist of sets of figures, graphics, etc. that are annotated with additional sets of properties of interest (metadata) stored in annotations files. *Regularized text data* consist of sets of sentences that share a similar structure, in the sense that similar entities or entities of the same type appear in the same general location in the sentence. This parallel structure allows for a "universal form" to be extracted, such that each sentence in the set is an instantiation of the universal form. The entities in each sentence have relationships between them that are specified in triple sheets and can be encoded as KG edges. More detailed descriptions of these data types can be found in Section IV.

IV. The BUILD-KG Workflow

We now introduce our proposed domain-agnostic BUILD-KG workflow for integrating heterogeneous data into a knowledge graph *(KG)* that could enable richer analytics not supported by the sources taken in isolation. In Sections IV-A–IV-C we present procedures for constructing KGs from *spreadsheet data*, *images with annotations*, and *regularized text data*, respectively. Then, in Section IV-D we outline the procedure for combining data in multiple formats into a unified KG.

A. Converting Spreadsheet Data into the KG Format

We present here construction of a KG from *spreadsheet data*, the first data type that we consider in this work. In the proposed domain-agnostic conversion procedure, we require the data to be in the format described in Section IV-A1. The output KG and its ontology are discussed in Section IV-A2, and the conversion procedure is presented in Section IV-A3.

1) The Input Data: The proposed procedure accepts spreadsheets in a specific format, which can be arrived at in collaboration with domain scientists, see Section VI for an illustration. The data in the desired format are stored in two spreadsheets: (1) a *data sheet* D, and (2) a *triple sheet* T. The data sheet D is similar to a relation, in that each row in D corresponds to a single data object. Each cell represents an entity in the resulting KG; related entities appear in the same row and will have corresponding edges in the resulting KG. If the same entity name appears in multiple rows, then it will correspond to a single entity in the resulting KG. This situation arises when an entity contributes to multiple data objects, and therefore has relationships with other entities from multiple rows.

The triple sheet T has three columns, one for each of subject, predicate, and object. Each row of T describes a *triple type* in the resulting KG, that is, a $(sType, p, oType)$ triple, in which $sType, oType \in \mathcal{T}$ represent respectively the subject and object entity-types corresponding to column names in the data sheet D, and $p \in \mathcal{P}$ is the predicate indicating the relationship type between them. As such, the triple sheet T enumerates the relationships between the entities present in the data sheet D. See Section V-C for an illustration.

2) The Output KG: The KG $\mathcal{G} = (\mathcal{E}, \mathcal{T}, \phi, \mathcal{P}, \mathcal{L})$ returned by the conversion procedure of Section IV-A3 contains the data from the data sheet D according to the format specified

Algorithm 1: Converting spreadsheet data into a KG.

Input: Data sheet D and triple sheet T.
Output: A KG $\mathcal{G}$ containing the data from D in the format specified by T.

1: $\mathcal{G} \leftarrow \emptyset$; // initialize the KG $\mathcal{G}$
// Create a node of $\mathcal{G}$ for each entity:
2: **for** $row \in D$ **do**
3: **for** $column \in D$ **do**
4: $e \leftarrow D[row][column]$; $t \leftarrow column.name$;
5: $\mathcal{G} \leftarrow \mathcal{G} \cup$ create node of type t for e;
// Create all the edges of $\mathcal{G}$:
6: **for** $sType, p, oType \in T$ **do**
7: **for** $row \in D$ **do**
8: $s \leftarrow row[sType]$; $o \leftarrow row[oType]$;
9: $\mathcal{G} \leftarrow \mathcal{G} \cup$ create edge of type p from s to o;
10: **return** $\mathcal{G}$;

by the triple sheet T. The entity set $\mathcal{E}$ of $\mathcal{G}$ consists of all the unique entries in D, and the entity-types $\mathcal{T}$ of $\mathcal{G}$ are the column headers of D. The entity-type labeling function ϕ in $\mathcal{G}$ maps each entry in D to its corresponding column header, and the predicate set $\mathcal{P}$ consists of all the entries from the second column of T. Finally, the set of triples $\mathcal{L}$ consists of a single triple of each triple type $(sType, p, oType)$ in T for each data object d in D, where the subject and object are the entries from d in columns $sType$ and $oType$, and the predicate is p.

3) The Conversion Procedure: Once the data are in the proper format, they can be converted into a KG in a straightforward manner, as outlined in Algorithm 1. The algorithm takes as inputs the data sheet D and the triple sheet T formatted as specified in Section IV-A1, and outputs the KG $\mathcal{G}$ described in Section IV-A2. The procedure works in two stages: (i) creating the nodes of the KG $\mathcal{G}$, and (ii) creating the edges of $\mathcal{G}$.

First, to create all the nodes of $\mathcal{G}$, the algorithm extracts from the data sheet D (lines 2–5) each entity e along with its type t (line 4), and creates the corresponding node of type t in the KG $\mathcal{G}$ (line 5). If the Neo4j system [25] is used for storing and processing the KG data, this process can be implemented via the following Cypher [26] query:

$$\texttt{MERGE } (\texttt{n} : t) \texttt{ SET n.id} = e$$

In Cypher, the keyword `MERGE` is used to either identify an already existing node pattern in the graph, or to create a new node pattern if one does not exist. Thus, the above query will create a node n in $\mathcal{G}$ of type t with the unique identifier e only if a node corresponding to e does not already exist in $\mathcal{G}$. In this way, Algorithm 1 guarantees that only a single node will be created for each unique entity in the data sheet D.

Next, to create all the edges in the KG $\mathcal{G}$, Algorithm 1 makes a triple of each triple type in T for each row in the data file D. To this end, the algorithm loops through the rows of T (lines 6–9), where each row specifies a subject type $sType$, a predicate p, and an object type $oType$. Then, the algorithm loops through the rows of D (lines 7–9), extracting from each row the subject s and the object o from the $sType$ and $oType$ columns (line 8). Finally, it creates an edge in the KG $\mathcal{G}$ of type p from s to o (line 9). Line 9 can be implemented using the following Cypher query:

$$\texttt{MATCH } (\texttt{n}_1), (\texttt{n}_2)$$
$$\texttt{WHERE n}_1\texttt{.id} = s \texttt{ AND n}_2\texttt{.id} = o$$
$$\texttt{MERGE } (\texttt{n}_1) - [\texttt{r} : p] \rightarrow (\texttt{n}_2)$$

This query first identifies the nodes $\texttt{n}_1$ and $\texttt{n}_2$ corresponding to the subject s and object o, and then creates an edge with the predicate type p from $\texttt{n}_1$ to $\texttt{n}_2$.

Finally, Algorithm 1 returns the KG $\mathcal{G}$ that has been populated with the data from the input data sheet D according to the format specified by the input triple sheet T (line 10).

B. Converting Images with Annotations into the KG Format

We now describe KG construction from *images with annotations*. To enable a domain-agnostic conversion procedure, we require that the annotations (image properties) be provided in a specific format described in Section IV-B1. The output KG and its ontology are discussed in Section IV-B2, and the conversion procedure is presented in Section IV-B3.

1) The Input Data: For this data type, the input consists of (1) a set I of one or more image files, and (2) an annotations file A. The image data in I can be in any of the popular image formats, e.g., JPG or PNG. The annotations file A consists of a single row for each image file in I. The column names in A are the property names provided by the annotations, and the row entries are the corresponding property values for the image. While the properties and values can be customized, for uniformity we recommend using consistent property names for similar images. See Section V-D for an illustration.

2) The Output KG: After executing the conversion procedure, see Section IV-B3, the resulting KG $\mathcal{G} = (\mathcal{E}, \mathcal{T}, \phi, \mathcal{P}, \mathcal{L})$ contains the set of images I and the data from the annotations file A. The entity set $\mathcal{E}$ of $\mathcal{G}$ consists of the images in I, along with all the property values in A. The entity types $\mathcal{T}$ are `Image` and the column headers of A, and the entity-type labeling function ϕ maps each image to `Image` and each property value to its corresponding column header. The predicate set $\mathcal{P} = \{$ `has_property` $\}$. The set $\mathcal{L}$ consists of triples (s, `has_property`, o), where s is an image from I and o is a property value from the corresponding row in A.

It is possible that some use cases may require additional triples to be present in the output KG $\mathcal{G}$, specifically, triples that relate property values to one another. In this case, the desired relationships can be encoded in a triple sheet T^* according to the spreadsheet data format specified in Section IV-A1, with T^* serving as an optional third input.

3) The Conversion Procedure: This conversion procedure takes as inputs the set of images I and the annotations file A described in Section IV-B1, and outputs the KG $\mathcal{G}$ described in Section IV-B2. To construct $\mathcal{G}$, the procedure adapts the inputs I and A to the spreadsheet data format described in Section IV-A1, and then invokes Algorithm 1.

To adapt the files in I and A to the spreadsheet data format, the procedure adds to A a column with the `Image` header. The values in this column are the image-file names for each row. The resulting annotations file A' is the data sheet D that is input into Algorithm 1. For the triple sheet T that is also required as an Algorithm 1 input, the procedure generates a sheet with one row for each column in the annotations file A; the subject of the row is `Image`, the predicate is `has_property`, and the object is the column name from A.

As discussed in Section IV-B2, it is possible that additional triples may be desired. In this case, the optional triple sheet T^* generated for the extra triples can be concatenated with the triple sheet T before invoking Algorithm 1.

Once the data sheet D and the triple sheet T have been generated, they are passed as inputs to Algorithm 1, which generates the KG $\mathcal{G}$ outlined in Section IV-B2.

C. Converting Regularized Text Data into the KG Format

We now describe construction of a KG from *regularized text data*. To handle the lack of uniformity present in textual data, we require that the text be regularized as described in Section IV-C1, to facilitate accurate triple extraction via named entity recognition (NER) [16]. The output KG and its ontology are discussed in Section IV-C2, and the conversion procedure, including the use of NER, is presented in Section IV-C3.

1) The Input Data: The input data consist of (1) a set S of regularized sentences, some of which are tagged sentences $S^T \subset S$, and (2) a triple sheet T formatted as described in Section IV-A1. *Regularized* means that the sentences share a similar structure. E.g., consider the sentences:

- Fig. 1 shows the RMSD for the interaction between 2HP and E. coli in clean water.
- Fig. 3 shows the radius of gyration for the interaction between dihydrogen phosphate and E. coli in multi-ion water.

The shared structure can be specified as "<Figure> shows the <Parameter> for the <Process> between <Ion> and <Protein> in <Solvent>."

Despite recent advances in NER performance, it still has limitations, especially with domain-specific terminology [27]. Therefore, we require that the input sentences be regularized this way to maximize the performance of the NER step of Section IV-C3, resulting in higher-quality KGs. We do not expect such regularization to be burdensome to domain scientists, as they may already tend to write sentences with similar structures in their work. We have confirmed this expectation with the scientists who provided the data for our use case.

We require some of the sentences $S^T \subset S$ to be tagged, to serve as the training data for the NER model. By using an NER model, we relieve domain scientists from the arduous task of tagging all the sentences in S. Tagging NER data consists of assigning a label to each token (word) in the text indicating the named entity that the token corresponds to. Entities in the sentence that consist of multiple tokens, e.g., *clean water*, have the first token labeled as `B-tokenName`, which stands for the "beginning" of the entity, and the subsequent tokens labeled as `I-tokenName`, which stands for the "inside" of the entity. The tag `O` (other) is used for any token that does not correspond to a named entity in the text.

The triple sheet T used in the procedure is in the format described in Section IV-A1. The entries in the subject and object columns of T must correspond to named entity types from the tagged sentences. See Section V-E for an example.

2) The Output KG: The conversion procedure of Section IV-C3 outputs a KG $\mathcal{G} = (\mathcal{E}, \mathcal{T}, \phi, \mathcal{P}, \mathcal{L})$ that contains the named entities from the sentences S connected by the relationships specified in the triple sheet T. The entity set $\mathcal{E}$ consists of the named entities in S, and the entity types $\mathcal{T}$ are the tags from the tagged sentences (aside from `O`). The entity-type labeling function ϕ maps each named entity to the tag assigned by the NER model, and the predicate set $\mathcal{P}$ consists of all the entries from the second column of the triple sheet T. The set of triples $\mathcal{L}$ consists of a single triple of each possible triple type $(sType, p, oType)$ in T for each sentence in S, where the subject and object are the $sType$- and $oType$-tagged entities in the sentence, and the predicate is p.

3) The Conversion Procedure: The procedure takes as inputs the set of sentences S, with some $S^T \subset S$ tagged, and the triple sheet T, see Section IV-C1, and outputs a KG $\mathcal{G}$, see Section IV-C2. To construct $\mathcal{G}$, an NER model is trained on S^T and used to extract named entities from S. Then, Algorithm 1 is invoked on the result of converting the outputs to the spreadsheet-data format described in Section IV-A1.

If the set $S^T \subset S$ is too small for successful training of a NER model, then S^T can be synthetically enlarged by automatically generating many sentences with similar structure. This generation can be done by substituting entity tokens in the regularized sentence structure with random words from a dictionary provided by the domain scientists. We used this technique to enlarge the training data set for our use case.

The trained model is used to provide tags for the remaining sentences in $S \backslash S^T$. Then, a data sheet D is generated from the complete set of tagged sentences S, just as for the spreadsheet data format, see Section IV-A1. The column headers of D are the NER tags (aside from `O`), and each row corresponds to a sentence in S, with the `t`-tagged entities in S appearing in the columns of D with column header `t`. The resulting data sheet D and the triple sheet T are passed as inputs to Algorithm 1, which returns the output KG $\mathcal{G}$.

D. Assembling a KG from Multiple Data Types and Data Sets

We now discuss the scenario in which a KG is created by combining data of multiple types. The KG can be constructed by using our proposed procedures for the corresponding data types, see Sections IV-A–IV-C. The data from individual sources can be converted one source at a time into a single unified KG, by executing Algorithm 1 on each source and using the same KG $\mathcal{G}$ for all the runs. To ensure that the data align and connect properly, one must pay particular attention to the terminology used across the sources. Terms representing the same entity/concept should be unified across the sources, such that the `MERGE` queries executed during the

runs of Algorithm 1 will identify pre-existing entities when appropriate instead of creating new entities, see Section IV-A3 for a discussion of the semantics of `MERGE` in Cypher.

In this scenario, we recommend maintaining data provenance when constructing the KG. Keeping data provenance is useful in general, but when assembling a KG from multiple data sources, it can be imperative down the line, e.g., for data cleaning or verification. Thus, we recommend maintaining as provenance the specific data set (input files) from which each KG triple originated. This can be achieved by adding the `provenance` property to each triple that would store the name of the data set from which the triple originated. The Cypher-query implementation of line 9 of Algorithm 1 can be modified to accomplish this by adding an appropriate `SET` clause.

In addition to ensuring that the same nodes are used when entities are shared across the input data sets, domain scientists might also choose to add to the KG extra edges connecting nodes across the converted source data, to express extra relationships based on domain semantics. Our proposed BUILD-KG workflow makes this possible with additional inputs. Domain scientists can input a set of specific (s, p, o) triples that would connect entities s and o via predicate p across the converted source data. These triples can be added to the KG using the command of line 9 in Algorithm 1.

V. The Use Case with Materials Science Data

In this section we outline our implementation of the proposed BUILD-KG workflow, and illustrate its application to a use case in the materials-science *(MS)* domain. Section V-A describes the tools used in our implementation of the BUILD-KG workflow, and Section V-B presents the source data for the use case. Sections V-C–V-E describe our instantiations of the workflow for the use-case MS *spreadsheet data*, *images with annotations*, and *regularized text data*, respectively. Finally, Section V-F describes the process of assembling a single unified KG from the three use-case data sets.

A. Tools Used in our Implementation of BUILD-KG

Implementing the BUILD-KG workflow requires a graph data-management system (DBMS), a graph query language, and a programming language. In our implementation, we used the Neo4j graph DBMS [25] with the Cypher graph query language [26]. The workflow was implemented in Python [28], including its publicly available package `py2neo` [29] for connecting to Neo4j and executing Cypher queries within Python code. For the named entity recognition (NER) model, we used the `flair` [24] package in Python. As training the NER model requires a data corpus and a tagger, for our use case we built the corpus by using the `ColumnCorpus` object in `flair`, with `SequenceTagger` from `flair` as the tagger. To train the `SequenceTagger`, we used the popular word-embedding model GloVe [30].

B. The Source Data in the Materials Science Use Case

As a use case for testing the proposed BUILD-KG workflow, we used data provided by researchers in the Science and Technologies for Phosphorus Sustainability (STEPS) Center. STEPS is sustainability driven, with a focus on creating systems that would allow successful capture of phosphates from a variety of environments. To build such systems, one needs to consider materials with high affinity to phosphate binding, low toxicity, and amenability to being easily cleaned or destroyed. Possible solutions include soft materials, such as bio-inspired proteins found in plants or microorganisms.

In our use-case data, the Escherichia coli (E. coli) phosphate-binding protein was used as a test model. This protein was investigated in various environments for its capability to bind with several forms of phosphate ions. We used data sets generated by experimental and computational E. coli tests. Merging the data sets into a KG would enable domain scientists to identify "junction points" and direct links between the data that may not be easily identifiable as the data sets grow in sizes. This would allow domain scientists to perform deeper analyses of the phosphate-binding properties of E. coli.

The data sets generated by STEPS scientists are as follows:

1) Data Set 1: Spreadsheet Data: This data set was obtained via conducting a series of experiments with the E. coli protein. The data consist of ultraviolet–visible (UV-Vis) spectrophotometer measurements summarized in spreadsheets. These measurements include absorbance, through which one can derive the phosphate capture rate via comparisons to the baseline. This series of experiments measured the capture rate under several conditions: temperature gradient, pH variations, ionic strength variations (concentration of the salt ions KCl present in the test solution), as well as in several system configurations: different amounts of the E. coli protein introduced in the test solution, different dihydrogen phosphate ion concentrations, several water matrix models, different types of nanoparticles used as the carrier, and various numbers of cycles in which the proteins were used.

2) Data Set 2: Images with Annotations: This data set was generated via analysis and summary of the raw data from Data Set 1 described in Section V-B1. It contains various graphs and charts, as well as data on the process kinetics and related isotherms, which were derived from the raw data points.

3) Data Set 3: Regularized Text Data: The data in this set were derived from atomistic simulations of the E.coli phosphate-binding protein carried out via all-atom molecular dynamics (MD) simulations. In the simulations, the test protein interacted with various phosphate ion types, and several values were computationally observed at the atomistic level, including binding rate, binding strength, speed of binding, and the particular regions of the protein affected by the presence of ions. This approach provides some insight into the kinetics of the binding process and supplements the experimental results.

The data for the MD simulations consist of input files (systems configurations, scripts, etc.), output data (files representing the dynamics of the systems across the simulation), and summary data (parameters derived from the systems via analysis). We used the summary data, which include various figures, together with their captions and descriptions.

C. Constructing a KG from the MS Spreadsheet Data

In this section we discuss our experience of constructing a KG from the *spreadsheet data* in our MS use case. Those data come from Data Set 1, see Section V-B1 for the details.

1) The Input Data: To arrive at the required input format specified in Section IV-A1, we used the data sheet D provided by Data Set 1, and generated a triple sheet T with the help of domain scientists. Table I shows a fragment of the data sheet D, which contains data about the experimental trials outlined in Section V-B1. For each trial, there is a single data object (row) containing information describing the trial, e.g., the trial ID, the protein used, and the solution used. For example, the first row of Table I details the trial *Temp1.1*. This trial used the protein *E. coli* on solution *Temp 01*, which contained *clean water* and *orthophosphate* at the original concentration of 1.12 *mg/L*; the trial used the temperature setting of $10°C$ and resulted in the absorbance of 0.234 *nm*.

Table II shows a fragment of the triple sheet T, which specifies the triple types desired in the resulting KG. Each row contains the subject, predicate, and object for a single triple type. E.g., the first row specifies the triple type (*Trial*, `used_protein`, *Protein*), with the meaning that each *Trial* and the corresponding *Protein* are linked by the `used_protein` relationship.

2) The Output KG: The inputs D and T determine both the ontology and the contents of the resulting KG $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{T}_1, \phi_1, \mathcal{P}_1, \mathcal{L}_1)$. The components of the KG $\mathcal{G}_1$ resulting from the inputs outlined in Section V-C1 are as follows. The entity set $\mathcal{E}_1$ consists of all the unique entries in D, that is, $\mathcal{E}_1 =$ { *Temp1.1*, *Temp1.2*, *Temp1.3*, *E. coli*, *Temp 01*, *clean water*, *orthophosphate*, 10, 0.234, 0.240, 0.226, 1.12 }. The entity-types $\mathcal{T}_1$ are the column headers of D, so $\mathcal{T}_1 =$ { `Trial`, `Protein`, `Solution`, `Solvent`, ..., `Original Conc. (mg/L)` }. The entity-type labeling function ϕ_1 maps each entry in D to its corresponding column header, e.g., ϕ_1 : *Temp1.1* $\rightarrow$ `Trial` and ϕ_1 : *Temp 01* $\rightarrow$ `Solution`. The predicate set $\mathcal{P}_1$ consists of all the entries from the second column of T, i.e., $\mathcal{P}_1 =$ { `used_protein`, `used_environment`, ..., `contains` }. Finally, the set of triples $\mathcal{L}_1$ consists of a single triple of each triple type $(sType, p, oType)$ in T for each data object d in D, where the subject and object are the entries from d in columns $sType$ and $oType$, and the predicate is p. For example, (*Temp 1.1*, `used_protein`, *E. coli*) $\in \mathcal{L}_1$.

3) The Conversion Process: We implemented the spreadsheet-conversion procedure of Algorithm 1, see Section IV-A3, in Python [28] using the `py2neo` [29] package to connect to our graph database stored in Neo4j [25]. Using the data sheet D and the triple sheet T of Section V-C1 as its inputs, Algorithm 1 returned the KG $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{T}_1, \phi_1, \mathcal{P}_1, \mathcal{L}_1)$ described in Section V-C2.

D. Constructing a KG from the MS Images with Annotations

In this section we discuss our experience of constructing a KG from the *images with annotations* in our MS use case. The data come from Data Set 2, see Section V-B2 for the details.

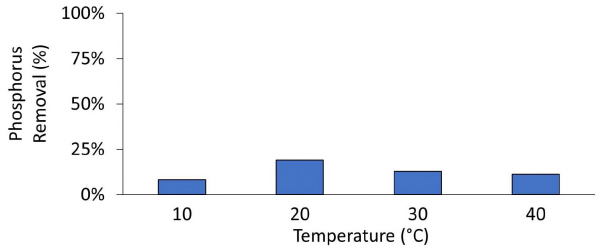

Fig. 2. An image from Data Set 2 described in Section V-B2.

1) The Input Data: To align with the required input format specified in Section IV-B1, we used the set of images I provided by Data Set 2, and compiled all the annotations into a single annotations file A with one row of annotations per file in I. Figure 2 shows a sample image from the set I of images in Data Set 2; this is a graph generated by MS researchers to summarize their experimental results. Table III shows the annotations for the image of Figure 2, which comprise a single line in the annotations file A. The annotations, i.e., properties of interest of the image, include in this case *title*, *x-axis*, *y-axis*, *description*, and *meaning*. E.g., the first annotation indicates that the image *title* is "Removal vs. Temperature."

2) The Output KG: The components of the KG $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{T}_2, \phi_2, \mathcal{P}_2, \mathcal{L}_2)$ resulting from the inputs discussed in Section V-D1 are as follows. The entity set $\mathcal{E}_2$ consists of the images in I and the unique entries in A, that is, $\mathcal{E}_2 =$ { *Figure 2*, *Removal vs. Temperature*, *temperature*, *removal rate (%)*, ... }. The entity types $\mathcal{T}_2$ are `Image` and the column headers of A: $\mathcal{T}_2 =$ { `Image`, `title`, `x-axis`, `y-axis`, `description`, `meaning` }. The entity-type labeling function ϕ_2 maps each image to `Image` and each property value to its corresponding column header, e.g., ϕ_2 : *Figure 2* $\rightarrow$ `Image` and ϕ_2 : *temperature* $\rightarrow$ `x-axis`. The predicate set $\mathcal{P}_2 =$ { `has_property` }. The set of triples $\mathcal{L}_2$ consists of triples $(s,$ `has_property`, $o)$, where s is an image from I and o is a property value from the corresponding row in A. E.g., (*Figure 2*, `has_property`, *temperature*) $\in \mathcal{L}_2$.

3) The Conversion Process: Following the procedure of Section IV-B3, we converted the set of images I and the annotations file A of Section V-D1 into the spreadsheet data format, obtaining a data sheet D and a triple sheet T. The adaptation procedure, described in Section IV-B3, was implemented as a Python [28] script. Table IV shows the data sheet D generated for the inputs I and A, see Section V-D1. D consists of the image annotations A with an additional column indicating the image corresponding to the annotations. Table V shows a fragment of the triple sheet T, which contains a single row for each column in the annotations file A. We passed these inputs D and T into our implementation of Algorithm 1 of Section V-C3, which returned the KG $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{T}_2, \phi_2, \mathcal{P}_2, \mathcal{L}_2)$ described in Section V-D2.

E. Constructing a KG from the MS Regularized Text Data

In this section we discuss our experience of constructing a KG from *regularized text data* in our MS use case. These data

TABLE I

FRAGMENT OF THE DATA SHEET FROM DATA SET 1 (SEE SECTION V-B1) INGESTED WITH THE DATA IN TABLE II BY THE CONVERSION PROCEDURE OF SECTION IV-A3 FOR SPREADSHEET DATA. EACH ROW DESCRIBES A SINGLE DATA OBJECT, AND THE COLUMNS REPRESENT DIFFERENT ENTITIES.

Trial	Protein	Solution	Solvent	Ion	Temperature ($°C$)	Absorbance (nm)	Original Conc. (mg/L)
Temp1.1	E. coli	Temp 01	clean water	orthophosphate	10	0.234	1.12
Temp1.2	E. coli	Temp 01	clean water	orthophosphate	10	0.240	1.12
Temp1.3	E. coli	Temp 01	clean water	orthophosphate	10	0.226	1.12

TABLE II

FRAGMENT OF THE TRIPLE SHEET FOR DATA SET 1 (SEE SECTION V-B1) INGESTED WITH THE DATA IN TABLE I BY THE CONVERSION PROCEDURE OF SECTION IV-A3 FOR SPREADSHEET DATA. EACH ROW DESCRIBES A TRIPLE TYPE OF THE FORM $(subjectType, p, objectType)$.

Subject	Predicate	Object
Trial	`used_protein`	Protein
Trial	`used_environment`	Solution
Protein	`used_on`	Solution
Protein	`had_property`	Absorbance
Solution	`contains`	Solvent
Solution	`contains`	Ion

come from Data Set 3, see Section V-B3 for the details.

1) The Input Data: To align with the required input format of Section IV-C1, we used the set S of regularized sentences from Data Set 3, tagging some of the sentences $S^T \subset S$. For example, the sentence "Figure 1 shows the RMSD for the interaction between 2HP and E. coli in clean water" in S has been converted to the following sequence of (Token, Tag) pairs in S^T: [(Figure, B-Figure), (1, I-Figure), (shows, O), (the, O), (RMSD, B-Parameter), (for, O), (the, O), (interaction, B-Process), (between, O), (2HP, B-IonMD), (and, O), (E, B-ProteinMD), (coli, I-ProteinMD), (in, O), (clean, B-SolventMD), (water, I-SolventMD)]. The tags indicate the named entities in the sentence and their types. E.g., the first two tokens in the example form a named entity of type Figure, while the third and fourth tokens are not named entities.

We also generated a triple sheet T with the help of domain scientists; a fragment of T is shown in Table VI. The triple sheet specifies the triple types desired in the resulting KG. Each row of the sheet contains the subject, predicate, and object for a single triple type. For example, the first row denotes the triple type (*Figure*, `illustrates`, *Process*), meaning that each *Figure* and the corresponding *Process* have an `illustrates` relationship between them.

2) The Output KG: The components of the KG $\mathcal{G}_3 = (\mathcal{E}_3, \mathcal{T}_3, \phi_3, \mathcal{P}_3, \mathcal{L}_3)$ resulting from the sample inputs provided in Section V-E1 are as follows. The entity set $\mathcal{E}_3$ consists of the named entities in S, that is, $\mathcal{E}_3 = \{$ *Figure 1*, *RMSD*, *interaction*, *2HP*, *E. coli*, *clean water* $\}$. The entity types $\mathcal{T}_3$ are the tags from the tagged sentences (aside from `O`), so $\mathcal{T}_3 = \{$ `Figure`, `Parameter`, `Process`, `IonMD`, `ProteinMD`, `SolventMD` $\}$. The entity-type labeling function ϕ_3 maps each named entity to the tag that it is assigned by the trained NER model, e.g., ϕ_3 : *RMSD* $\rightarrow$ `Parameter` and ϕ_3 : *2HP* $\rightarrow$ `IonMD`. The predicate set $\mathcal{P}_3$ consists of all the entries from the second column of the triple sheet T, i.e., $\mathcal{P}_3 = \{$ `illustrates`, `measured_by`, `involves`, `occurs_in` $\}$. Finally, the set of triples $\mathcal{L}$ consists of a single triple of each possible triple type $(sType, p, oType)$ in T for each sentence in S, where the subject and object are the $sType$- and $oType$-tagged entities in the sentence, and the predicate is p. For example, (*interaction*, `measured_by`, *RMSD*) $\in \mathcal{L}_3$.

3) The Conversion Process: Following the conversion procedure of Section IV-C3, we first trained a named entity recognition (NER) model on the set of tagged sentences S^T synthetically enhanced by automatically generating sentences with similar structure, see Section V-E1 for an example. See Sections V-A and IV-C3 for more details about our NER model and the process for generating synthetic sentences.

Next, we used the trained model to provide tags for the sentences $S \setminus S^T$. Then we adapted the complete set of tagged sentences S to the format of the data sheet D used for the spreadsheet data input (see Section IV-A1). To this end, we implemented the adaptation procedure described in Section IV-C3 as a Python [28] script. Table VII shows the data sheet D that was generated for the sample sentence and tags given in Section V-E1. D consists of a single row for the sentence; each entry is a named entity from the sentence, and each named entity appears in the column corresponding to its tag (type).

Finally, we passed the newly generated data sheet D and the triple sheet T (see Section V-E1) to our implementation of Algorithm 1 of Section V-C3. The run of Algorithm 1 returned the KG $\mathcal{G}_3 = (\mathcal{E}_3, \mathcal{T}_3, \phi_3, \mathcal{P}_3, \mathcal{L}_3)$ described in Section V-E2.

F. Assembling a KG from Multiple Data Types and Data Sets

For our use case, it would be valuable to the domain scientists if the KGs $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$ were unified into a single KG. As discussed in Section IV-D, this can be accomplished by building a single KG $\mathcal{G}$ using the BUILD-KG conversion process for each data type and each data set. Executing the conversion processes of Sections V-C3, V-D3, and V-E3 resulted in the KG $\mathcal{G}$ with the contents of the KGs $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$ described in Sections V-C2, V-D2, and V-E2, respectively, with $\mathcal{G}$ having the unified ontology of the three KGs. Further, each node or edge that is shared between $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$ was mapped by BUILD-KG into the same node or edge in $\mathcal{G}$.

Following the recommendations given in Section IV-D, we consulted domain scientists to ensure that the three data sets in our use case would use common terminology where appropriate. Additionally, we maintained data provenance when assembling $\mathcal{G}$. To this end, each triple (edge) t in $\mathcal{G}$ has a property $t.$`provenance` that stores the ID of one of the three source data sets from which that triple originated.

The domain scientists were also looking for additional relationships that would connect entities across the data sets,

TABLE III

THE ANNOTATIONS FROM DATA SET 2 DESCRIBED IN SECTION V-B2 FOR THE IMAGE SHOWN IN FIGURE 2. THE ANNOTATIONS HAVE BEEN USED AS INPUTS TO THE CONVERSION PROCEDURE OF SECTION IV-B3 FOR IMAGES WITH ANNOTATIONS.

title	x-axis	y-axis	description	meaning
Removal vs. Temperature	temperature	removal rate (%)	Removal rate as a function of temperature (the temperature-dependence of removal)	The amount of phosphate that is removed at different temperatures

TABLE IV

THE DATA SHEET GENERATED WHEN EXECUTING THE CONVERSION PROCEDURE OF SECTION IV-B3 FOR IMAGES WITH ANNOTATIONS; THE PROCEDURE USED AS INPUTS THE IMAGE SHOWN IN FIGURE 2 AND ITS ANNOTATIONS SHOWN IN TABLE III.

Image	title	x-axis	y-axis	description	meaning
Figure 2	Removal vs. Temperature	temperature	removal rate (%)	Removal rate as a function of temperature (the temperature-dependence of removal)	The amount of phosphate that is removed at different temperatures

TABLE V

FRAGMENT OF THE TRIPLE SHEET GENERATED WHEN EXECUTING THE CONVERSION PROCEDURE OF SECTION IV-B3 ON THE IMAGE OF FIGURE 2 AND ITS ANNOTATIONS SHOWN IN TABLE III.

Subject	Predicate	Object
Image	`has_property`	title
Image	`has_property`	x-axis
Image	`has_property`	y-axis

TABLE VI

FRAGMENT OF THE TRIPLE SHEET FOR THE DATA SET 3 OF SECTION V-B3, USED AS AN INPUT TO THE CONVERSION PROCEDURE OF SECTION IV-C3 FOR REGULARIZED TEXT DATA. EACH ROW DESCRIBES A TRIPLE TYPE OF THE FORM $(subjectType, p, objectType)$.

Subject	Predicate	Object
Figure	`illustrates`	Process
Process	`measured_by`	Parameter
Process	`involves`	IonMD
Process	`involves`	ProteinMD
Process	`occurs_in`	SolventMD

thus enabling richer analytics on $\mathcal{G}$ than what the data sets or even $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$ could allow in isolation. To this end, the scientists provided a set of (s, p, o) triples to connect such entities; s and o would come from different data sets but have the relationship p between them in $\mathcal{G}$. An example of such a triple is (*2HP*, `simulates`, *orthophosphate*), where *2HP* is a node of type `IonMD` in Data Set 3 (see Section V-E1) and *orthophosphate* is a node of type `Ion` in Data Set 1 (see Section V-C1). The triple, see Figure 1, indicates that the `IonMD` *2HP* is used to simulate the `Ion` *orthophosphate* in molecular-dynamics simulations. We added the triples to $\mathcal{G}$ via the Cypher query implementing line 9 of Algorithm 1.

These steps completed our BUILD-KG conversion and unification process for the use case. We provided the resulting KG $\mathcal{G}$ to the STEPS scientists for their further use and exploration.

TABLE VII

THE DATA SHEET GENERATED BY THE CONVERSION PROCEDURE OF SECTION IV-C3 FOR REGULARIZED TEXT DATA FOR THE SAMPLE TAGGED SENTENCE OF SECTION V-E1 FROM THE DATA SET 3 OF SECTION V-B3.

Figure	Parameter	Process	IonMD	ProteinMD	SolventMD
Figure 1	RMSD	interaction	2HP	E. coli	clean water

TABLE VIII

SAMPLE SEMANTIC SENTENCES GENERATED BASED ON A SEMANTIC DESCRIPTION OF DATA SET 1 PRESENTED IN SECTION V-B1.

Defines	Semantic sentence
Entity-type	E. coli is a Protein.
Entity-type	Water containing orthophosphate is a Solution.
Predicate	An experimental trial resulted in 0.234 nm of absorbance.
Predicate	A solution had an original concentration of 1.12 mg/L.
Predicate	The protein E. coli was used on solution Temp 01.

VI. HUMANS-IN-THE-LOOP IN THE WORKFLOW

We now detail the preprocessing steps that should be taken in order to properly format the data that are to be integrated by our proposed BUILD-KG workflow. This preprocessing involves close collaboration with domain scientists as humans-in-the-loop, and is needed for achieving full understanding of their data and for ensuring that the data are captured in the knowledge graph *(KG)* as accurately as possible.

To facilitate productive collaboration with domain scientists, we recommend the following four data-preprocessing steps:

1) Obtain raw data from the domain scientists, along with semantic descriptions of the data elements;
2) Generate sample semantic *subject-predicate-object* sentences based on the semantic descriptions;
3) Ask the domain scientists to validate or correct the semantic sentences; and
4) Based on the semantic sentences, format the raw data such that they would meet the formatting requirements presented in Sections IV-A1, IV-B1, and IV-C1.

Examples of the data received from the STEPS scientists in Step 1 are shown in Figs. 1(a)-(b). From these data, we generated semantic *subject-verb-object* sentences to show the scientists (Step 2). The second column of Table VIII shows some sample semantic sentences that were generated based on the semantic descriptions of the spreadsheet data from Data Set 1 (see Section V-B1) given by the scientists. The first two sentences in Table VIII are used to define entity-types in the resulting KG. They take the general form "`<EntityName>` is a `<EntityType>`." The last three sentences in Table VIII are used to define predicates in the resulting KG. They take the general form "`<Subject> <Predicate> <Object>`."

After the generation of the semantic sentences, domain scientists validate them in Step 3 to ensure that the data are understood and captured correctly before the KG construction. Based on the entity-type sentences, the scientists can verify the entities selected from the data sets and the types assigned to the entities. Based on the predicate sentences, the scientists can verify the related entities and the relationship names. Verifying and correcting these items enables the scientists to control the data in the resulting KG and its ontology, so that it would align with their expectations and needs. In this step, we found it useful to generate with the STEPS scientists a dictionary of domain-specific terminology, to enable us to align the terminology used across the data sets. The dictionary allowed us to build the ontology more efficiently. It also cleared up some misunderstandings about the terms used. E.g., instead of using a single node of type *Solution* for the liquids used in the experiments, the scientists requested that we use nodes of types *Solvent* and *Ion* to separate the contents of the liquids.

Steps 2 and 3 can be repeated as many times as necessary to produce an integrated KG that satisfies the needs of the domain scientists. Such a KG will have a proper structural and semantic representation of the original data, and will provide insightful connections (junction points) across the data. Once this iterative process is done, in Step 4 the raw data can be reformatted to match the BUILD-KG input-format requirements. In our use case, the semantic sentences shown in Table VIII were used to generate the spreadsheet data-type inputs shown in Tables I and II and discussed in Section V-C1.

VII. Conclusion

In this paper we introduced a domain-agnostic workflow called BUILD-KG for integrating heterogeneous scientific and experimental data from multiple sources into a single unified KG that can enable richer data analytics. BUILD-KG is broadly applicable, accepting input data in popular structured and unstructured storage formats. It is also designed to be carried out with end users as humans-in-the-loop, which makes BUILD-KG domain aware. We presented the BUILD-KG workflow, reported on our experiences with applying it to data in the materials-science domain, and provided suggestions on involving domain scientists in BUILD-KG as humans-in-the-loop. We posit that our proposed BUILD-KG workflow can enable domain scientists to seamlessly convert their data into the KG format, unify their data with those shared by other domain scientists, and then apply to the resulting KG data-analysis tasks, such as rule mining and machine learning, that may not be supported by the data sources taken in isolation. In this way, the BUILD-KG workflow can make KGs more accessible to domain scientists, thus encouraging greater use and exploration, while increasing collaboration and richness of research results. Our future work includes expanding the proposed collection of conversion procedures within BUILD-KG to include other popular data-storage formats.

References

[1] A. Bosselut, H. Rashkin *et al.*, "COMET: Commonsense Transformers for Automatic Knowledge Graph Construction," 2019.

[2] J. L. Martinez-Rodriguez, I. Lopez-Arevalo, and A. B. Rios-Alvarado, "OpenIE-based approach for Knowledge Graph construction from text," *Expert Syst. Appl.*, vol. 113, pp. 339–355, 2018.

[3] X. Wu, J. Wu *et al.*, "Automatic Knowledge Graph Construction: A Report on the 2019 ICDM/ICBK Contest," in *IEEE ICDM*, 2019, pp. 1540–1545.

[4] C. Wang, X. Ma *et al.*, "Information extraction and knowledge graph construction from geoscience literature," *Comput. Geosci.*, vol. 112, pp. 112–120, 2018.

[5] P. Chen, Y. Lu *et al.*, "An automatic knowledge graph construction system for K-12 education," in *ACM L@S*, 2018, pp. 1–4.

[6] Y. Luan, L. He *et al.*, "Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction," 2018.

[7] L. Li, P. Wang *et al.*, "Real-world data medical knowledge graph: construction and applications," *Artif Intell Med*, vol. 103, p. 101817, 2020.

[8] F.-L. Li, H. Chen *et al.*, "AliMeKG: Domain Knowledge Graph Construction and Application in E-commerce," in *ACM CIKM*, 2020, pp. 2581–2588.

[9] H. Huang, Z. Hong *et al.*, "Knowledge Graph Construction and Application of Power Grid Equipment," *Math. Probl. Eng.*, vol. 2020, pp. 1–10, 2020.

[10] S. Elhammadi, L. V.S. Lakshmanan *et al.*, "A High Precision Pipeline for Financial Knowledge Graph Construction," in *ICCL*, 2020, pp. 967–977.

[11] X. Hao, Z. Ji *et al.*, "Construction and Application of a Knowledge Graph," *Remote Sens.*, vol. 13, no. 13, p. 2511, 2021.

[12] L. Asprino, E. Daga *et al.*, "Knowledge Graph Construction with a *Façade*: A Unified Method to Access Heterogeneous Data Sources on the Web," *ACM TOIT*, vol. 23, no. 1, pp. 1–31, 2023.

[13] H. Zhang, X. Wang *et al.*, "SAKA: an intelligent platform for semi-automated knowledge graph construction and application," *Serv. Oriented Comput. and Appl.*, vol. 17, no. 3, pp. 201–212, 2023.

[14] M. Kejriwal, *Domain-Specific Knowledge Graph Construction*. Springer International Publishing, 2019.

[15] H. Ye, N. Zhang *et al.*, "Generative Knowledge Graph Construction: A Review," 2022.

[16] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investig.*, vol. 30, no. 1, pp. 3–26, 2007.

[17] S. Pyysalo, J. Kanerva *et al.*, "WikiBERT models: Deep transfer learning for many languages," in *NoDaLiDa*, 2021, pp. 1–10.

[18] J. Devlin, M. Chang *et al.*, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[19] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *CoNLL*, 2003, pp. 142–147.

[20] J. Lee, W. Yoon *et al.*, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[21] M. Sung, M. Jeong *et al.*, "BERN2: an advanced neural biomedical named entity recognition and normalization tool," *Bioinformatics*, vol. 38, no. 20, pp. 4837–4839, 2022.

[22] N. Walker, A. Trewartha *et al.*, "The impact of domain-specific pre-training on named entity recognition tasks in materials science," *Available at SSRN 3950755*, 2021.

[23] T. Gupta, M. Zaki *et al.*, "MatSciBERT: A materials domain language model for text mining and information extraction," *npj Comput. Mater.*, vol. 8, no. 1, p. 102, 2022.

[24] A. Akbik, T. Bergmann *et al.*, "FLAIR: An easy-to-use framework for state-of-the-art NLP," in *NAACL*, 2019, pp. 54–59.

[25] The Neo4j Team, "The Neo4j operations manual v4.4," 2022.

[26] ——, "The Neo4j Cypher manual v4.4," 2022.

[27] M. Marrero, J. Urbano *et al.*, "Named Entity Recognition: Fallacies, challenges and opportunities," *Comput. Stand. Interfaces*, vol. 35, no. 5, pp. 482–489, 2013.

[28] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.

[29] N. Small, "The py2neo v4 handbook," 2019.

[30] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.