

Phenotype Knowledge Translator (PheKnowLator): A FAIR Ecosystem for Representing Large-Scale Biomedical Knowledge

This is an initial draft – manuscript is still progress

Tiffany J. Callahan MPH, PhD¹; Ignacio J. Tripodi, PhD²; Adrienne L. Stefanski, PhD¹; Luca Cappelletti, BS³; Sanya B. Taneja BS⁴; Jordan M. Wyrwa DO⁵; Elena Casiraghi, PhD³; Nicolas A. Matentzoglou⁶; Justin Reese, PhD⁷; Felipe Mello, MS⁸; Jonathan C. Silverstein, MD⁹; Charles Tapley Hoyt, PhD¹⁰; Richard D. Boyce, PhD⁹; Scott A. Malec, PhD⁹; Deepak R. Unni, MS¹¹; Marcin P. Joachimiak, PhD⁷; Christopher J. Mungall, PhD⁷; Giorgio Valentini, PhD¹²; Nicole A. Vasilevsky, PhD¹³; Robert Hoehndorf, PhD¹⁴; Tellen D. Bennett, MD¹⁵; Michael G. Khan, MD PhD¹⁵; William A. Baumgartner Jr. PhD^{1†}; Lawrence E. Hunter PhD^{1†}

[†]Shared Senior Authorship; *Corresponding Author

¹Computational Bioscience Program, University of Colorado Anschutz Medical Campus, Aurora, CO 80045, USA

²Computer Science Department, Interdisciplinary Quantitative Biology, University of Colorado Boulder, Boulder, CO 80031, UAS

³Computer Science, Università degli Studi di Milano, Milan, 20122, Italy

⁴Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA

⁵Department of Physical Medicine and Rehabilitation, School of Medicine, University of Colorado Anschutz Medical Campus, Aurora, CO 80045, USA

⁶Semanticly Ltd, London, United Kingdom

⁷Division of Environmental Genomics and Systems Biology, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

⁸Computer Science Program, Georgia Institute of Technology, Philadelphia, PA 19118, USA

⁹Department of Biomedical Informatics, University of Pittsburgh School of Medicine, Pittsburgh, PA 15260, USA

¹⁰Laboratory of Systems Pharmacology, Harvard Medical School, Boston, MA 02115, USA

¹¹Genome Biology Unit, European Molecular Biology Laboratory, Heidelberg, 69117, Germany

¹²AnacletoLab, Dipartimento di Informatica, Università degli Studi di Milano, Milan, 20122, Italy

¹³Translational and Integrative Sciences Lab, University of Colorado Anschutz Medical Campus, Aurora, CO 80045, USA

¹⁴Computer, Electrical and Mathematical Sciences & Engineering Division, Computational Bioscience Research Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudi Arabia

¹⁵Department of Pediatrics, Sections of Informatics and Data Science and Critical Care Medicine, School of Medicine, University of Colorado School of Medicine, Aurora, CO 80045, USA

ORCID^s (ordered by author)

0000-0002-8169-9049, 0000-0003-3066-4273, NA, 0000-0002-1269-2038, 0000-0003-1707-1617, 0000-0002-5455-5859, 0000-0003-2024-7572, 0000-0002-7356-1779, 0000-0002-2170-2250, 0000-0003-0332-4249, 0000-0002-9252-6039, 0000-0003-4423-4370, 0000-0002-2993-2085, 0000-0003-1696-1781, 0000-0002-3583-7340, 0000-0001-8175-045X, 0000-0002-6601-2165, 0000-0002-5694-3919, 0000-0001-5208-3432, 0000-0001-8149-5890, 0000-0003-1483-4236, 0000-0003-4786-6875, 0000-0001-6717-5313, 0000-0003-1455-3370, 0000-0001-6717-5313, 0000-0002-5455-5859, 0000-0003-1455-3370

ABSTRACT

Although knowledge graphs (KGs) are used extensively in the biomedical domain to model complex phenomena. Unfortunately, existing KG construction methods often lack standardization, have limited scope, and scale poorly as the size of input data sources increase. PheKnowLator (Phenotype Knowledge Translator), a fully customizable semantic ecosystem for Findable, Accessible, Interoperable, and Reusable (FAIR) construction of ontologically-grounded KGs, was developed to address these problems. Pheknowlator resolves the lack of standardization using ontologies and the FAIR principles. Limited scope and poor scaling are resolved by enabling the construction of multiple types of KGs and using distributed computation. PheKnowLator was evaluated in two ways. First, a survey of existing open-source KG construction methods available on GitHub was performed. The survey identified 14 existing open-source KG construction methods and found that the PheKnowLator Ecosystem was comparable to the other methods in terms of its KG construction functionality, maturity, availability, usability, and reproducibility. Next, the KGs of human disease mechanisms were built by applying PheKnowLator to 12 Open Biological and Biomedical Foundry Ontologies and 31 publicly available resources using 15 edge types. The computational performance of the KG builds was recorded. The resulting KGs varied significantly in size from 737,556 nodes and 5,487,821 edges with 293 unique edge types to 15,903,225 nodes and 47,420,725 edges with 847 unique edge types. Computational performance varied by build step such that on average, the data download step used the least resources (3.5 min; 7.9 GiB) and the KG construction step used the most resources (319.6 min; 119.7 GiB). The performance also varied by KG build; the subclass-based build with inverse relations and OWL-NETS transformation took the longest time and used the most memory (615.9 min; 147.1 GiB). **PLACEHOLDER FOR EMBEDDING EXPERIMENT RESULTS**. Although relatively new, PheKnowLator's user base is growing rapidly supporting projects focused on toxicogenomic mechanistic inference, human disease causation, and pharmacokinetic natural product-drug interaction. Preliminary results demonstrate that the PheKnowLator Ecosystem is one of the first fully customizable open-source KG construction frameworks able to provide a wide range of functionality without compromising performance or usability.

INTRODUCTION

The worldwide growth of biomedical data is exponential with the volume of molecular data alone and it is expected to surpass more than four Exabytes by 2025 [1]. Rapid advancements in next-generation sequencing technology and personal health data produced from the continual sharing of data via wearable devices and social networking have made tremendous amounts of diverse data available for secondary use [2,3]. Multimodal data like these capture different views and, when properly combined, help characterize complex systems [4]. Unfortunately, these data are often highly distributed and heterogeneous, can be difficult to access due to licensing restrictions, lack interoperability, and often have inconsistent underlying models or representations, which limits most researchers from fully utilizing and/or combining them [5,6].

Knowledge graphs have frequently been used to systematically interrogate the biology underlying complicated systems, organisms, and diseases [7]. Graph representation learning algorithms have been developed to analyze biomedical KGs, ranging from random walk-based homogeneous graph embedding methods [8] to graph neural networks expressly designed to manage heterogeneous data [9]. In the biomedical domain, KGs are usually constructed from the combination of a wide range of data sources like Linked Open Data, ontologies, the literature, data derived from electronic health records, and multi-omics experiments [6,10]. Although KGs are extensively used in the biomedical domain, we have found that many KG construction methods remain largely unable to account for the use of different standardized terminologies or vocabularies, which are often difficult to use, and perform poorly as the size of the resulting KG increases in scale [6,10].

To solve these problems, we developed PheKnowLator (Phenotype KNowledge Translator), an ecosystem of tools for FAIR construction of ontologically-grounded KGs. To address challenges with integrating data from different standards, PheKnowLator provides a suite of tools, built on the FAIR Data Principles (i.e., Findable, Accessible, Interoperable, and Reproducible; [11]). To improve usability, PheKnowLator includes several Jupyter Notebooks with step-by-step examples of how to use the tools provided in the Ecosystem. PheKnowLator also includes an interactive script that walks users through the steps needed to compile the different input dependencies required to build a KG. Finally, in terms of scalability, PheKnowLator is built on Ray (<https://www.ray.io/>), providing users with the ability to parallelize KG builds. The rest of this paper is organized as follows. First, we provide a detailed definition of a KG. Next, we provide a review of existing KG construction methods. Then, we describe the unsolved barriers and challenges to automating KG construction. The paper concludes with the introduction of PheKnowLator.

What is a Knowledge Graph?

In the simplest terms, a graph can be thought of as an undirected, unweighted network $G(N, L)$, where N is the set of nodes and L is the set of observed edges between these nodes. In the biomedical context, nodes usually represent different kinds of biological entities like genes, proteins or diseases. Edges or relations are used to specify different types of relationships (e.g., interaction, substance that treats) that can exist between a pair of nodes. A triple can then be defined as a node-relation-node statement (e.g., geneA - interacts with - geneB). Please see

[Supplemental Table 1](#) for definitions of concepts introduced throughout the manuscript.

At the state of the art, multiple definitions of KGs have been proposed [12–14], all sharing the understanding that KGs are more than simple large-scale graphs; indeed, we believe all the various definitions may be best summarized by Ehrlinger's and Wöb's (2016) KG definition: "A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge." [12]. We extend Ehrlinger's and Wöb's (2016) definition and consider a KG a graph-based data structure representing a variety of heterogeneous entities (i.e., nodes) and multiple types of relationships between them and serving as an "abstract framework" that can infer new knowledge to address a variety of applications and use cases. Within the scope of our extended definition, we include a wide variety of KGs; KGs constructed using a highly formal approach to representation where every entity and relationship is modeled as assertions and axioms in a formal logic such as a description logic like the Web Ontology Language (OWL; [15]), grounded in ontological primitives, and suitable for logical reasoning [16]. An example would be KaBOB (Knowledge Base of Biomedicine [KaBOB]; [5]). There are also KGs that are constructed with much less logical formality such as those represented solely in RDF and RDFSchemas [17] (e.g., KG-COVID-19 [18], DisGeNET [19], OpenBioLink [20], Bio2RDF [21] and those that do not use a formal standard like Hetionet [22], SPOKE [23], SemMedDB [24]. An example of a KG designed to represent the Central Dogma in Homo sapiens is shown in [Figure 1](#).

Methods to Construct Biomedical Knowledge Graphs

Researchers have made substantial efforts to develop novel methods to construct, integrate, and evaluate KGs, as highlighted in recent reviews [6,10]. Regardless of the specific approach utilized to construct a KG, most methods apply the following steps, which are ultimately driven by the use case: (1) **Resource Identification**. Identify the data sources that will be used for nodes and edges. Most biomedical KGs are built from data that have been extracted or transformed from an existing KG (e.g., SPOKE [23]), the literature (e.g., SemMedDB [24]) or most commonly from data database/Linked Open Data resources (e.g., KG-COVID-19 [18], DisGeNET [19], OpenBioLink [20], Bio2RDF [21], and Hetionet [22]); (2) **Data Integration**. Transformation and/or preprocessing of resources from the prior step into an edge list or set of nodes and edges. This usually consists of procedures to verify the quality of the ingested data, normalize node and edge identifiers, and extract and format needed node and edge metadata (e.g., node and edge labels, and edge weights); and (3) **KG Construction**. Combine edges and/or nodes into a KG (which usually follows an existing framework, model, or knowledge representation schema).

As the base building blocks, Steps 1 to 3 tend to be implemented using a mix of manual and automatic techniques. While an extensive review of the techniques within each of these methods is not within the scope of this project, in the following we recall the main characteristics and examples of manual and automatic KG construction techniques. An overview of each method type is described below.

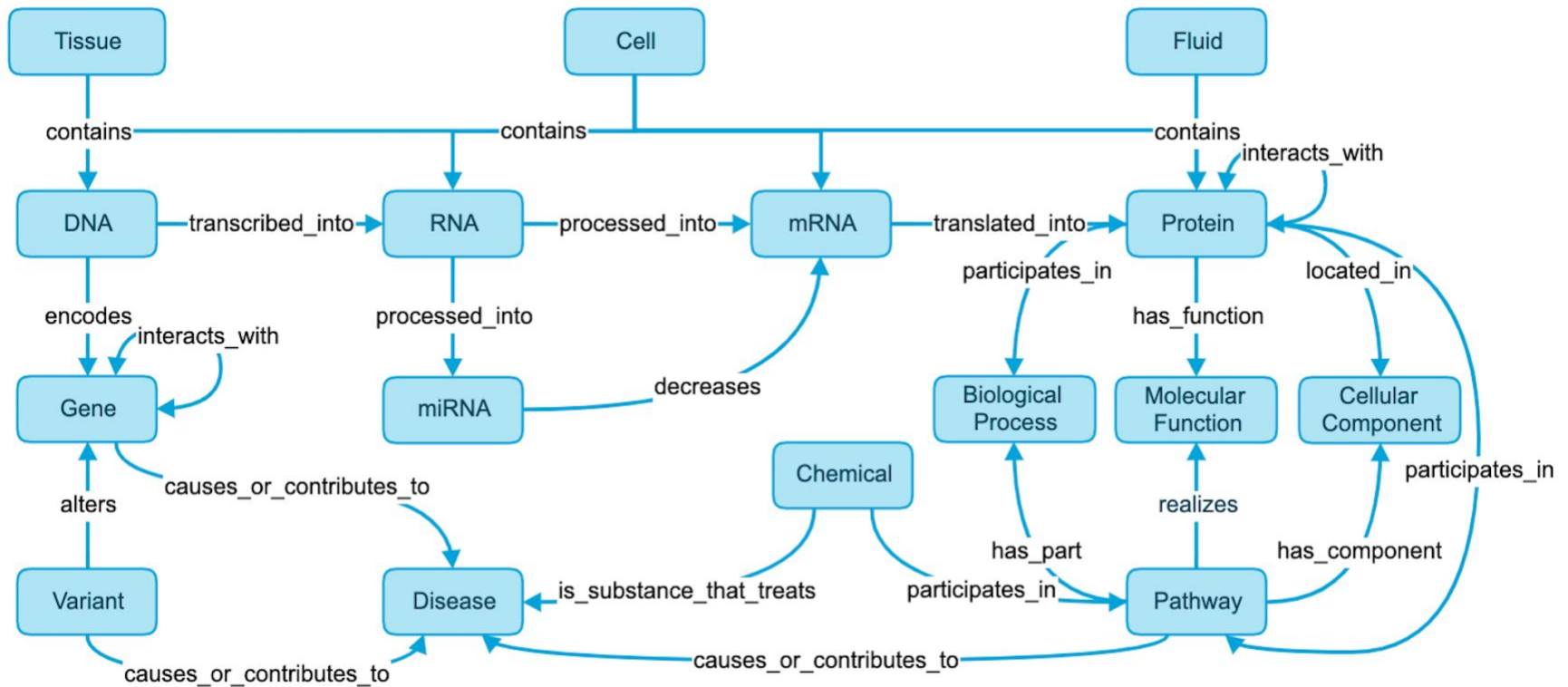


Figure 1. A Knowledge Graph Representation of the Central Dogma in the Context of Human Disease.

This KG is an attempt to provide a simple overview of our currently accepted knowledge of the Central Dogma, where anatomical entities like tissues, cells, and fluids contain genomic entities like DNA, RNA, mRNA, and proteins. DNA encodes genes that can interact with each other. Genes can also be altered by variants and cause disease. Finally, proteins can also interact with each other and participate in pathways and biological processes.

Manual KG Construction

Manual methods usually consist of some form of human curation, parsing, and/or extraction of specific data from existing sources. In some cases, manual curation has also been used to construct a brand-new resource [citation?]. Manually constructing a KG includes identifying the nodes that will comprise an edge as well as selecting the, possibly multiple, relations that will connect them (e.g., if the nodes are the gene entity [A2M \(NCBIGene:2; https://www.ncbi.nlm.nih.gov/gene/2\)](https://www.ncbi.nlm.nih.gov/gene/2) and [Alzheimer's Disease \(HP:0002511; https://hpo.jax.org/app/browse/term/HP:0002511\)](https://hpo.jax.org/app/browse/term/HP:0002511), the relation [Causes or Contributes to \(RO:0003302; http://purl.obolibrary.org/obo/RO_0003302\)](http://purl.obolibrary.org/obo/RO_0003302) could be used to connect them). In addition to identifying nodes and selecting relations between them, manual construction efforts will usually also include metadata for each edge (e.g., an identifier that describes who created the edge).

The community-curated collection of Open Biomedical Ontologies (OBOs) is a prime example of manually constructed KGs. Built to the standards and principles governed by the OBO Foundry [25,26] is an open-source collaborative that provides standards and principles for the construction of biomedical ontologies (e.g., Human Phenotype Ontology [27], Human Disease Ontology [28], Protein Ontology [29]). While many of the ontologies within the OBO Foundry are verified using reasoners, which can also be used to derive novel assertions (or novel, logical inferred, triples, which are logically inferred), most of these ontologies are manually constructed using rigorous processes and through the collaboration of domain experts. Gene Ontology Causal Activity Models (GO-CAMs; [30]) are another example of a manually constructed KG, which are built on top of the Gene Ontology [29] by a team of professional ontologists and/or biocurators.

The primary benefit of using a manual approach to construct KGs is that the resulting structure is much less likely to contain unpredicted inconsistencies, errors, or other sources of variance, since humans are naturally able to detect and correct them. On the other hand, manual KG construction is a time-demanding task, which may be prone to human errors, especially when the volume of data is large and when we lack clear standardization rules. The manual construction of a high-quality KG requires the collaboration of multiple domain experts and can take a significant amount of time to complete and resources to maintain. In this case, when the KG being constructed becomes large, inter- and intra-curator variability may result in inconsistencies due to human errors. For these reasons, without a well-organized protocol and a dedicated community, manual construction approaches often lack the scalability to stay up to date with the speed at which data and knowledge are generated in the biomedical domain.

Automatic KG Construction

Like manual construction methods, automatic methods to construct KGs must also identify nodes and relations and can also provide metadata. The core functionality of most automated techniques consists of algorithms that: (1) extract entities and relations from both structured and unstructured data sources [32]; (2) align or normalize extracted entities to a standard set of identifiers (i.e., using custom mapping frameworks, an existing gold standard corpus, or through the use of other KGs or ontologies [6]); and (3) construct the final KG by combining the

extracted entities and relations. These algorithms are usually either supervised, semi-supervised rule-based, or fully unsupervised. Supervised techniques require access to labeled data and leverage classical machine learning techniques and deep learning models [10]. Examples of supervised methods in the Bioinformatics field are those developed to identify protein-protein [33–36], disease-gene [37–40], and drug-target and/or drug-disease [41–47] relations. Semi-supervised rule-based approaches are heavily influenced by work done in the Natural Language Processing (NLP) domain and include tools which help reduce the initial space manual experts are required to verify (e.g., Argo [48], BioQRator [49], PubAnnotation [50], TextPresso Central [51], CurEx [52], LocText [53]). Unsupervised methods aim to remove manual curation efforts altogether and as a result must make inferences directly from the data. Also rooted deeply in the NLP domain, these techniques often heavily rely on thresholding from scores derived from techniques like co-occurrence [54–60], link prediction [61–63], clustering [64], and entity similarity [65–69]. For a detailed review of some of these techniques see [6,10].

An example of a KG constructed using semi-supervised rule-base techniques is MedTruth [70]. This KG is constructed from a pre-trained model that has been designed to distinguish reliable data from untrustworthy data. Once trained, the model is combined with a subset of existing knowledge (in the form of triples) and used to score new triples via their similarity to trustworthy triples. COMET [71] and NormCo [72] are examples of KGs constructed using fully unsupervised techniques. COMET or COMmonsEense Transformers derive KGs from transferring knowledge from pre-trained language models (built from rich text sources) into common sense KGs [71]. NormCo uses entity embeddings to train recurrent neural networks designed to predict specific types of co-mention sequences learned via distantly supervised priors derived from the BioASQ [73] dataset [72].

The primary benefits of using automated KG construction methods are scalability and maintainability. By relying on existing and trusted sources of existing relationships, these approaches do not require the same amount of resources as manual curation. While these methods have better scalability, as fundamentally statistical approaches they are subject to false positives and false negatives and are only as good as the methods they are trained against or the gold standards they are verified with.

Challenges to Constructing Knowledge Graphs

Current barriers and challenges to fully automating the construction of large-scale KGs within the biomedical domain include: (i) requiring substantial data preprocessing in order to address data quality and interoperability issues; (ii) a lack of agreed upon models or schemas for representing knowledge; and (iii) a lack of benchmark tasks to validate KGs and the methods to construct them.

Data Quality and Interoperability

Biomedical data, especially data that have been extracted from experiments or sources of free-text, are notoriously prone to data quality errors [74–78]. While it seems clear that there should be procedures to verify data quality in place when constructing KGs from biomedical resources, very few KG construction approaches include methods to verify data prior to constructing a KG. Biomedical data are represented using a wide range of standards and identifiers. Interoperability

can be improved through the use of ontologies and by following FAIR Data Principles (i.e., Findable, Accessible, Interoperable, and Reproducible; [11]). While many existing approaches to construct KGs use ontologies and some follow the FAIR principles, very few do both.

Knowledge Models

While there are many methods for constructing a KG, the best model for representing these data and the complex relationships between them is less clear [79,80]. Although not a comprehensive list, some of the models or approaches to represent knowledge that have been used in the biomedical domain include well-established models like BioPax [81] and the Biological Expression Language (BEL; [82]). A highly expressive Semantic Web standard for representing complex knowledge, OWL allows for expressive representation of existing knowledge and the generation of new knowledge via deductive inference [16]. By enforcing explicit semantics, OWL provides a common data representation language, which has shown promise when used to integrate large biomedical data [83]. Finally, there are also methods that sit on top of some of the prior described knowledge models or standards. For example, OWL-NETS [84] and Hetionet metapaths [22] are methods which have been developed to convert complicated representations of knowledge into simpler representations and have proven useful for tasks like link prediction and visualization. While there is no universal schema or model to represent biomedical data stored in a KG, there are also no existing studies which have compared all existing models (or even a subset of existing models). Understanding if, how, and when the different models for representing knowledge matter is an important area in need of exploration.

Validation Benchmarks

One of the biggest challenges to developing novel KG construction methods and to constructing novel KGs is knowing how to verify and validate them. Other types of algorithms within the biomedical domain, like link prediction and knowledge graph embedding, make use of well-established benchmarks like YAGO [85], DBPedia [86], and Wikidata [87]. While these benchmarks, which are not strictly biomedical in nature, can be used for validating methods designed for biomedical data, they cannot as easily be used to construct biomedical KGs. OpenBioLink [20] was developed as a benchmark for biomedical KGs, but exclusively for use in link prediction tasks. While it might not be possible to create a benchmark KG explicitly intended to verify or validate biomedical KG construction methods, development of a set of tasks (e.g., specific types of biomedical entity prediction or node classification) for verifying or validating biomedical KG content and construction would benefit the community. The current work does not claim to address this issue, but recognizes it is an important area of future research that we would like to invite others in the community to work on with us.

Objectives

Although KGs are used extensively in the biomedical domain, existing KG construction methods remain largely unable to account for the use of different standards and knowledge models used by the various relevant sources of knowledge. Current KG construction methods are also difficult to use and often perform poorly as the size of the resulting KGs increase in scale. To solve these problems, we developed PheKnowLator (Phenotype KNowledge Translator), an

ecosystem of tools for FAIR construction of ontologically grounded KGs. PheKnowLator KGs are fully customizable, enabling the use of alternative knowledge models, unidirectional or bidirectional relations, and with or without semantic property graph abstraction. In the following sections we first introduce the PheKnowLator Ecosystem. Then, we evaluate the Ecosystem using tasks designed to test its functionality and performance. A dedicated Zenodo Community has been established and provides access to software, presentations, and preprints related to this ecosystem (<https://zenodo.org/communities/pheknowlator-ecosystem>).

METHODS

Please see the [Supplemental Material](#) for definitions and acronyms used throughout the paper ([Supplemental Material Table 1](#)), source code, example workflows, and additional results from analyses presented in this paper.

The PheKnowLator Ecosystem

The PheKnowLator Ecosystem is built on the FAIR principles [11]. As illustrated in [Figure 2](#), with respect to *Findability*, PheKnowLator uses unique persistent identifiers for all downloaded and processed data, metadata documentation, generated reports, log files, and all cloud compute instances and containers. With respect to *Accessibility*, all data, documentation, and log files are accessible via RESTful API access to a dedicated Google Cloud Storage (GCS) Bucket. All builds are versioned on GitHub, Google’s Container Registry, and DockerHub. Finally, Jupyter Notebooks are created as a means for providing user-friendly access and to assist with running different portions of the KG build pipeline. With respect to *Interoperability*, PheKnowLator uses Semantic Web standards like OWL, is grounded in Open Biomedical Foundry ontologies (OBO), and, wherever possible, adopts standard identifiers, like Universal Resource Identifiers (URIs). Finally, with respect to *Reusability*, PheKnowLator provides extensive documentation of all code, output, and workflows. PheKnowLator also uses Semantic Versioning, is licensed ([Apache-2.0](#)) and includes documentation of the licensing constraints enforced for all ingested data sources. PheKnowLator adopts a standard Description Logics architecture [88] that defines a KG as $K = \langle T, A \rangle$, where T is the TBox and A is the ABox. The TBox is defined as the set of terminological axioms that represent the schema of a specific domain and is composed of classes and properties (i.e., relationships). TBox concepts and roles are defined with respect to the World Wide Web Consortium’s OWL 2 EL standard [89,90] and consist of concept names C_0, C_1, \dots (e.g., *neoplasm* [[HP:0002664](#)]), property names R_0, R_1, \dots (e.g., *determined by* [[RO:0002507](#)]), and constructors like $C_1 \sqcap C_2$ intersection (i.e., “and”; modeled using the [owl:intersectionOf](#) relation), $C_1 \sqcup C_2$ union (i.e., “or”; modeled using the [owl:unionOf](#) relation), $\neg C$ complement (i.e., “not”; modeled using the [owl:complementOf](#) relation), and R^{-1} inverse (modeled using the [owl:inverseOf](#) relation). Primitive concept axioms take the form of $C_1 \sqsubseteq C_2$, where C_1 is a concept name and C_2 is an EL concept. In this form, C_2 is *subsumed by* C_1 or C_2 is-a C_1 . The [rdfs:subClassOf](#) property is used for this type of axiom. The ABox is defined as the set of assertional axioms that represent the data used to make assertions about a specific domain and is composed of individuals or instances of TBox-defined classes and their attributes and roles. ABox assertions are defined as an instance of some TBox class $C(a)$ and roles are defined as an instance of





 Findable	Unique Persistent Identifiers <ul style="list-style-type: none"> • Data: Original and processed data • Metadata: Logs and quality reports • Infrastructure: Compute and containers
 Accessible	Publicly Available <ul style="list-style-type: none"> • Storage: RESTful access to builds • Builds: Versioned on Docker Hub • Notebooks: User-friendly examples
 Interoperable	Standardized Resources <ul style="list-style-type: none"> • Data: Ontology alignment • Metadata: Provenance reporting • Output: Standard file formats
 Reusable	Detailed Documentation <ul style="list-style-type: none"> • Releases: Code, data, builds • Versioning: Semantic versioning • Licensing: Internal/external resources

Figure 2. The PheKnowLator Ecosystem on FAIR Principles.

The PheKnowLator Ecosystem is built on the FAIR principles of Findability, Accessibility, Interoperability, and Reusability.

some TBox property $R(a, b)$, where a and b are two named individuals. The `rdf:type` property is used to instantiate or create an individual. Unlike other construction methods that we are aware of, PheKnowLator enables the construction of KGs using either a TBox or an ABox-based knowledge representation.

As shown in [Figure 3](#), the PheKnowLator Ecosystem consists of four primary components: (i) Data Download; (ii) Data Preparation; (iii) KG Construction; and (iv) Output Generation. Each of these components are described below.

Data Download

This component takes as input three text files¹ (examples of each file are provided in [Supplemental Material Figure 1](#)). The first two files contain lists of Universal Resource Links (URLs) for downloading ontologies ([ontology_source_list.txt](#)) and non-ontology data sources ([edge_source_list.txt](#)), such as Uniprot, Comparative Toxicogenomics Database, or data from

¹To reduce the burden of creating these documents and help users identify needed information, there is a command line script (https://github.com/callahantiff/PheKnowLator/blob/master/generates_dependency_documents.py).

the Human Protein Atlas. Within these documents, each resource is tagged with a keyword that indicates the type of entity it represents (e.g., “gene”, “disease”, “drug”). These keywords correspond to tuples or keyword pairs (e.g., “gene-disease”) in the third document ([resource_info.txt](#)), which provides instructions on how to assemble the edges for each entity referenced in the keyword tuples. The steps to construct each edge type are implemented in the *Data Preparation* component. Custom tools are available to automatically download each resource. During the download process, detailed metadata is recorded and stored for each resource ([downloaded_build_metadata.txt](#); July 2021 build in [Supplemental Material Figure 2](#)).

Data Preparation

This component processes all data from the *Data Download* component. For this process, there are scripts and Jupyter Notebooks that facilitate detailed workflows for preparing and processing ontology and non-ontology data. The ontology cleaning protocol corresponds to the Jupyter Notebook titled [Ontology_Cleaning.ipynb](#)². This notebook performs several steps to clean an individual ontology and/or a set of merged ontologies. After the processing is complete, an ontology cleaning report is output that documents statistics for each ontology pre- and post-processing ([ontology_cleaning_report.txt](#); an example from the July 2021 build is shown in [Supplemental Material Figure 3](#)). The non-ontology data source cleaning process also has a corresponding Jupyter Notebook ([Data_Preparation.ipynb](#)³). Currently, the workflow provides several steps for processing a wide range of Linked Open Data and is built to support PheKnowLators Monthly builds (described in detail below) as well as to provide examples for users who want to create their own pipelines. The current processing procedures include everything from identifier cross-mapping, entity metadata extraction, concept annotation, and filtering data. These procedures are purposefully defined to take advantage of existing public endpoints and Application Programming Interfaces (APIs) whenever possible. All processed data for the ontology and non-ontology cleaning protocols are documented in metadata reports, which are output at each processing stage ([preprocessed_build_metadata.txt](#); July 2021 build shown in [Supplemental Material Figure 4](#)).

Knowledge Graph Construction

This component consists of the following three stages:

1. **Build Edge Lists.** This step follows the processing instructions documented in the [resource_info.txt](#) document. The edge list procedure consists of applying filtering and evidence criteria, removing unneeded columns and duplicate values, mapping identifiers, and retrieving entity namespace types (i.e., “gene”, “disease”, “drug”). To facilitate these steps, a universal file parser is being developed, which when used in combination with the instructions in [resource_info.txt](#), is currently able to automatically process over 30 distinct file types. Once processing is complete, the edge lists are output as a hashtable or dictionary ([Master_Edge_List_Dict.json](#) [July 2021 build example]) and metadata for each processed edge data source are output to a document called [edge_source_metadata.txt](#) (July 2021 build shown in [Supplemental Material](#)

²Ontology Cleaning: https://github.com/callahantiff/PheKnowLator/blob/master/notebooks/Ontology_Cleaning.ipynb.

³Data Cleaning: https://github.com/callahantiff/PheKnowLator/blob/master/notebooks/Data_Preparation.ipynb.

[Figure 5](#)).

- 2. Merge Ontologies.** This step leverages OWLTools [91] (April 06, 2020 release) to merge all ontologies listed in the [ontology_source_list.txt](#). If the ontology cleaning protocol is followed, then this step is not needed. Similar to the edge resources, metadata for each ontology source is output to a document called [ontology_source_metadata.txt](#) (July 2021 build example shown in [Supplemental Material Figure 6](#)).
- 3. Select Hyperparameters.** This step executes the primary KG construction processes via configuration of three hyperparameters. (a) *Knowledge Model*. PheKnowLator is equipped with two knowledge representation approaches: ABox or *instance-based* [92] and TBox or *subclass-based* [5] ([Figure 4](#) provides an example of how these approaches differ). For the instance-based approach, non-ontology data are made from an instance of an existing ontology class. For the subclass-based approach, non-ontology data are made a subclass of an existing ontology class. Both approaches require the alignment of non-ontology entities to existing ontologies, which are stored as a dictionary ([subclass_construction_map.pkl](#)) and created as part of the processes described in the *Data Preparation* component. For additional examples, see the construction approach README⁴; (b) *Relation Strategy*. It is assumed that relations will be drawn from ontologies like the Relation Ontology (RO; <https://oborel.github.io/>), although this is not a requirement. The benefit of using an ontology for relations is that PheKnowLator is able to use a reasoner to infer inverse relations. Additional functionality is also provided to automatically generate inverse relations for one-way, implicitly symmetric relations for edge types that represent different kinds of biological interactions (e.g., molecular interactions); and (c) *Semantic Abstraction*. OWL is a highly expressive language, but its use comes at the cost of structurally complex KGs. Many of the triples or edges responsible for this complexity do not contain biologically meaningful information (e.g., OWL-encoded classes and axioms). To enable users the ability to create versions of PheKnowLator KGs that only include biologically meaningful edges (i.e., all edges representing OWL syntactic elements have been removed), the OWL-NETS method (v2.0⁵) [84], is made accessible through the PheKnowLator Ecosystem. An example of the OWL-NETS output for an OWL-encoded class is shown in [Figure 5](#).

As of release v2.1.0⁶, PheKnowLator also includes functionality to ensure that the resulting KG contains a single connected component, and all edges are “purified” or harmonized to the KG construction approach. To ensure that the KG contains a single connected component, a set of root nodes are derived by searching for each node's highest ancestor concept. If a node has no ancestors, its immediate neighbors are searched and the most frequently visited, highest common ancestor among its neighbors is selected. If none of the node's neighbors have any common ancestors, an ancestor concept is selected at random. If a node has more than one neighbor, the highest common ancestor concept among the neighbors is selected. Each root

⁴README: https://github.com/callahantiff/PheKnowLator/tree/master/resources/construction_approach#readme

⁵OWL-NETS v2.0: <https://github.com/callahantiff/PheKnowLator/wiki/OWL-NETS-2.0>

⁶PheKnowLator (pkt-kg) v2.1.0: <https://github.com/callahantiff/PheKnowLator/releases/tag/v2.1.0>

node is then added to the KG as [rdfs:subClassOf](#) or [rdf:type](#) of a user-provided class (defaults to [BFO:0000001](#) [*entity*]).

The purification or harmonization procedure ensures an OWL-NETS KG is consistent with the KG construction approach (i.e., subclass- or instance-based). For subclass-based builds, all edges containing [rdf:type](#) are updated to [rdfs:subClassOf](#) and all ancestors of the subjects included in the [rdf:type](#) edges are made [rdfs:subClassOf](#) of the corresponding object. For instance-based builds, all edges containing [rdfs:subClassOf](#) are updated to [rdf:type](#) and all ancestors of the subjects included in the [rdfs:subClassOf](#) edge are made [rdf:type](#) of the corresponding object.

Output Generation

In addition to the metadata files described in the prior components, node and relation metadata information are also output as a dictionary ([node_metadata_dict.pkl](#)) and a tab-delimited flat text file (**_NodeLabels.txt*). Additionally, PheKnowLator outputs each KG in a variety of different formats (see [Supplemental Material Table 2](#)). PheKnowLator also supports the generation of node embeddings and includes a public SPARQL Endpoint (<http://sparql.pheknowlator.com/>).

A Hub for Knowledge Graph Benchmarks

In addition to providing an algorithm to construct KGs, the PheKnowLator Ecosystem also supports monthly KG builds. These builds are implemented using dedicated Docker containers, which output data directly to a PheKnowLator GCS Bucket. See the build README for additional details (<https://github.com/callahantiff/PheKnowLator/tree/master/builds#readme>). The knowledge representation and data used for each monthly build are detailed on the Wiki (<https://github.com/callahantiff/PheKnowLator/wiki/v2.0.0>) and described in the Evaluation section below. Each build is extensively documented via log files, which are output to a timestamped PheKnowLator GCS Bucket associated with each build (i.e., data download and preprocessing steps [[pkt_builder_phases12_log.log](#)]; kg build steps [[pkt_build_log.log](#)]; both files are examples from the July 2021 build).

Technical Specifications

PheKnowLator was developed using Python 3.6.2 and is available through GitHub (<https://github.com/callahantiff/PheKnowLator>) and PyPI (<https://pypi.org/project/pkt-kg/>). For the *Data Preprocessing* and *KG Construction* components, PheKnowLator is automatically parallelized using Ray (v.1.1.1). The codebase is actively maintained using GitHub Actions-based continuous integration and includes a unit test coverage of 92% of the current codebase [as of August 2021]. PheKnowLator can be installed via pip install and run from the command line using the [Main.py](#) script or from the [main.ipynb](#) notebook. Scripts maintain custom builds through Docker and pre-built containers (v20.10.6) from DockerHub (<https://hub.docker.com/repository/docker/callahantiff/pheknowlator>). All monthly build data are publicly available through the PheKnowLator GCS bucket (<https://console.cloud.google.com/storage/browser/pheknowlator>).

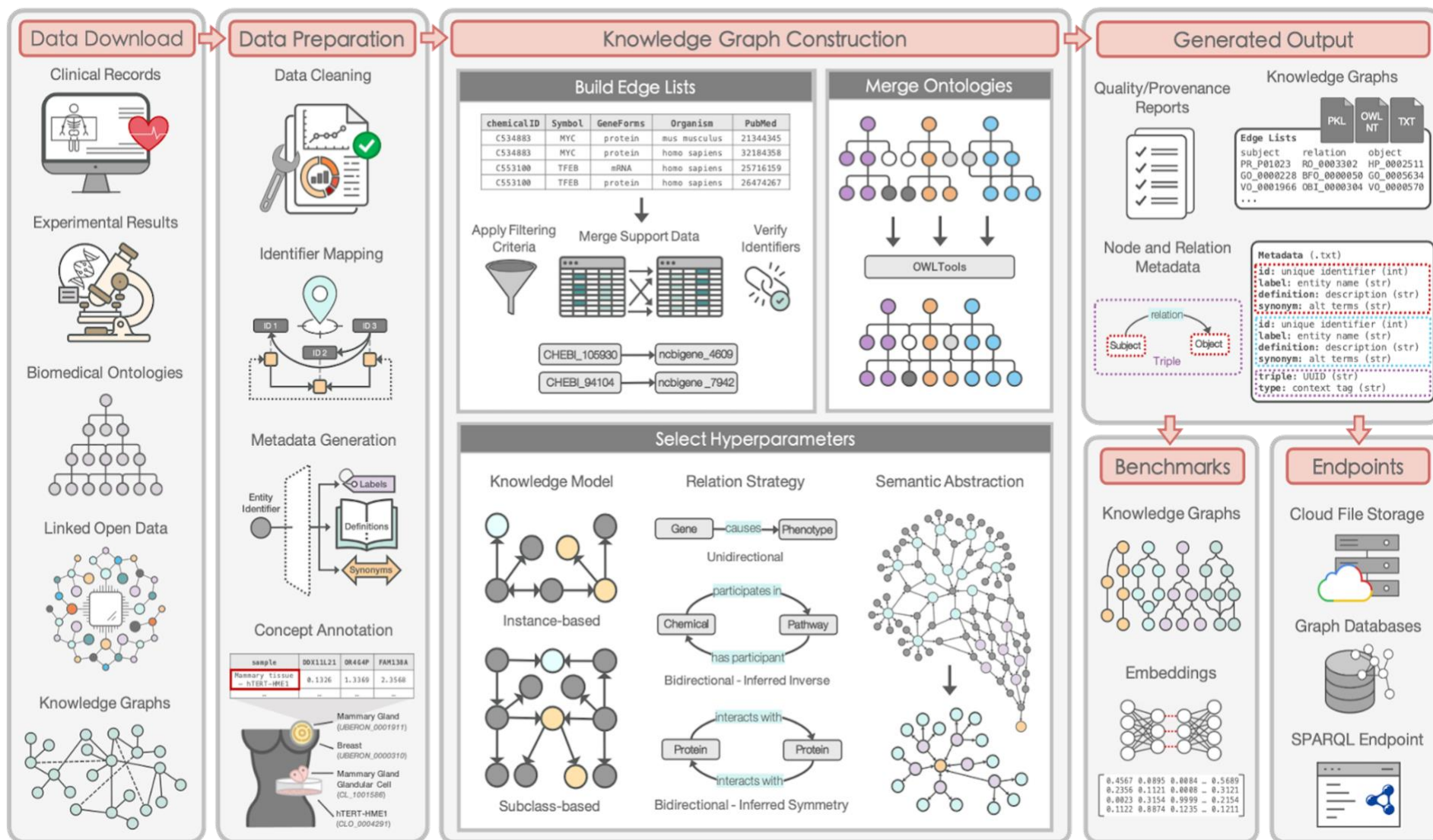


Figure 3. The PheKnowLator Ecosystem.

The PheKnowLator Ecosystem includes tools to download and prepare data, construct knowledge graphs, and generate a wide range of outputs. These outputs support the production of benchmarks and are accessible through public endpoints. Click the following link to access a larger version of this figure <https://bit.ly/3xZaKCI>. Acronyms - NT: N-Triples file format; OWL: Web Ontology Language; PKL: Python pickle file format; SPARQL: SPARQL Protocol and RDF Query Language.

A2M → Causes or Contributes to Condition → Alzheimer Disease	
A2M is a non-ontology class entity	
Gene is a class in the Sequence Ontology (so_0000704) needed to connect A2M to the Knowledge Graph	
Alzheimer Disease is a class in the Human Phenotype Ontology (HP_0002511)	
Instance-based	Subclass-based
<pre> A2M, rdfs:subClassOf, Gene A2M, rdf:type, owl:Class UUID1, rdf:type, A2M UUID1, rdf:type, owl:NamedIndividual UUID2, rdf:type, Alzheimer Disease UUID2, rdf:type, owl:NamedIndividual UUID1, causes_or_contributes_to_condition, UUID2 </pre>	<pre> A2M, rdfs:subClassOf, Gene A2M, rdf:type, owl:Class UUID1, rdfs:subClassOf, A2M UUID1, rdfs:subClassOf, UUID2 UUID2, rdf:type, owl:Restriction UUID2, owl:someValuesFrom, Alzheimer Disease UUID2, owl:onProperty, causes_or_contributes_to_condition </pre>

Figure 4. Construction Approach Hyperparameter.

This figure provides an example of the instance- and subclass-based construction approaches. Additional information can be found on GitHub in the construction approach README.md (<https://bit.ly/2W4lh1m>). Acronyms - HP: Human Phenotype Ontology; OWL: Web Ontology Language; RDF: Resource Description Framework; RDFS: Resource Description Framework Schema; SO: Sequence Ontology; UUID: Universally Unique Identifier.

Evaluation

PheKnowLator was evaluated in two ways: (i) survey and manual review of existing biomedical KG construction methods on GitHub and (ii) construction of human disease mechanism KGs, where the computational performance of each build was recorded, and descriptive statistics of the resulting builds was examined.

Open-Source Software Survey

A survey was performed to examine how PheKnowLator compares to existing open-source biomedical KG construction methods. To provide an unbiased comparison, no assumptions were made regarding a specific set of user requirements. Instead, the goal of the survey was to provide a detailed overview of existing methods. To assist with this survey, five criteria (adapted from [93]) were used to compare methods with respect to the task of constructing biomedical KGs, as shown in [Table 1](#). The criteria included: KG construction functionality, maturity, availability, usability, and reproducibility. Examples of questions used to assess each criterion are provided in the table. The full set of survey questions (n = 44) are available as a Google Form [94]. Existing open-source biomedical KG construction methods were identified by performing a keyword search against the GitHub API. The following words were combined to form 31 distinct keyword phrases which were queried against existing GitHub repository descriptions and README content:

Scenario 1: owl:unionOf constructor

```
<!-- http://purl.obolibrary.org/obo/CL_0000995 -->
<owl:Class rdf:about="http://purl.obolibrary.org/obo/CL_0000995">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://purl.obolibrary.org/obo/CL_0001021"/>
        <rdf:Description rdf:about="http://purl.obolibrary.org/obo/CL_0001026"/>
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/CL_0001060"/>
</owl:Class>
```

http://purl.obolibrary.org/obo/CL_0000995

The OWL class CL_0000995 (i.e. CD34-positive, CD38-positive common myeloid progenitor OR CD34-positive, CD38-positive common lymphoid progenitor) was built by taking the union:

- CL_0001021 (i.e. CD34-positive, CD38-positive common lymphoid progenitor)
- CL_0001026 (i.e. CD34-positive, CD38-positive common myeloid progenitor)

OWL-NETS would decode this class into:

```
CL_0001021, rdfs:subClassOf, CL_0000995
CL_0001026, rdfs:subClassOf, CL_0000995
CL_0000995, rdfs:subClassOf, CL_0001060
```

Scenario 2: owl:intersectionOf constructor

```
<!-- http://purl.obolibrary.org/obo/HP_0000340 -->
<owl:Class rdf:about="http://purl.obolibrary.org/obo/HP_0000340">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://purl.obolibrary.org/obo/BFO_0000051"/>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <rdf:Description rdf:about="http://purl.obolibrary.org/obo/PATO_0001481"/>
            <owl:Restriction>
              <owl:onProperty rdf:resource="http://purl.obolibrary.org/obo/RO_0000052"/>
              <owl:someValuesFrom rdf:resource="http://purl.obolibrary.org/obo/UBERON_0000200"/>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="http://purl.obolibrary.org/obo/RO_0002573"/>
              <owl:someValuesFrom rdf:resource="http://purl.obolibrary.org/obo/PATO_0000460"/>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/HP_0000290"/>
</owl:Class>
```

http://purl.obolibrary.org/obo/HP_0000340

The owl class HP_0000340 (i.e. sloping forehead) was built by taking the intersection of:

- PATO_0001481, RO_0000052, UBERON_0000200 (i.e. sloped, inheres in, forehead)
- PATO_0001481, RO_0002573, PATO_0000460 (i.e. sloped, has modifier, abnormal)

OWL-NETS would decode this class into:

```
HP_0000340, RO_0000052, PATO_0001481
HP_0000340, RO_0000052, UBERON_0000200
HP_0000340, RO_0002573, PATO_0000460
HP_0000340, rdfs:subClassOf, HP_0000290
```

Figure 5. OWL-NETS owl:Class Example.

This figure provides examples of how OWL-NETS decodes [owl:Class](#) entities constructed using [owl:intersectionOf](#) and [owl:unionOf](#). Acronyms - BFO: Basic Formal Ontology; CL: Cell Line Ontology; HP: Human Phenotype Ontology; OWL: Web Ontology Language; PATO: Phenotype And Trait Ontology; RDF: Resource Description Framework; RDFS: Resource Description Framework Schema; RO: Relation Ontology; UBERON: Uber-Anatomy Ontology.

"biological", "bio", "medical", "biomedical", "life science",
"semantic", "knowledge graph", "kg", "graph", "network",
"build", "construction", "construct", "create", "creation"

The GitHub scraper is a publicly available GitHub Gist and was run May 2020 (<https://gist.github.com/callahantiff/0ae1c00df9bec7228be3f6bda5466d73>). The survey was completed and verified in June 2021 by two researchers independently. The survey was scored out of a total score of 5 points, which was derived as the sum of the ratio of coverage out of 1 point for each survey category (i.e., KG Construction Functionality (10/11 questions scored); Availability (2/5 questions scored); Usability (9/10 questions scored); Maturity (5/5 questions scored); and Reproducibility (6/6 questions scored)).

Human Disease Mechanism Knowledge Graph Builds

A biologically meaningful KG representation, designed to model mechanisms of human disease, was developed through collaboration with a PhD-level molecular biologist⁷. KGs were constructed using 12 open biomedical ontologies, 31 Linked Open Data sets, and results from two large-scale experiments. Combining these sources facilitated the addition of the following edge types:

chemical-disease, chemical-gene, chemical-biological process,
chemical-cellular component, chemical-molecular function,
chemical-pathway, chemical-phenotype, chemical-protein, disease-
phenotype, gene-disease, gene-gene, gene-pathway, gene-phenotype,
gene-protein, gene-rna, biological process-pathway, pathway-cellular
component, pathway-molecular function, protein-anatomy, protein-
catalyst, protein-cell, protein-cofactor, protein-biological process,
protein-cellular component, protein-molecular function, protein-
pathway, protein-protein, rna-anatomy, rna-cell, rna-protein, variant-
disease, variant-gene, variant-phenotype.

Additional details on the data sources used to construct these KGs can be found on the project Wiki (<https://github.com/callahantiff/PheKnowLator/wiki/v2-Data-Sources>). All data were downloaded and processed on May 1, 2021 (May 2021 build GCS Bucket: <https://bit.ly/3ATj7I5>). The 12 different parameterizations of PheKnowLator KGs were constructed to demonstrate the range of KGs that PheKnowLator can produce. These KGs were built by varying the construction approach (i.e., instance- versus subclass-based), relation strategy (i.e., using only relations versus relations with their inverse), and whether OWL-encoded classes were decoded [with and without construction approach harmonization].

Using these 12 builds, two forms of evaluation were applied: (i) **Computational Performance**. Performance metrics evaluated when building the PheKnowLator KGs included the total runtime (minutes) and minimum, maximum, and average memory use (GiB); and (ii) **Descriptive**

⁷The knowledge representation developed as part of this collaboration is used to create the knowledge graphs generated each month. For more details see: <https://github.com/callahantiff/PheKnowLator/tree/master/builds>.

Table 1. Open-Source Knowledge Graph Construction Tool Survey.

Criteria	Description	Example Question
Construction Functionality	Evaluated by examining how well the method covers the steps needed to construct a knowledge graph from downloading and processing data and building edge lists to generating and outputting a KG	Is there functionality to download data? Can multiple types of KGs be constructed? Is preprocessing or filtering performed as part of the construction process?
Maturity	Evaluated by examining the level, stage or development phase of a method	Is a versioning system in place? Have many releases been made? Are procedures in place to enable collaboration?
Availability	Evaluated by examining the openness of a method and the ease of obtaining a copy of the method	Is the method licensed? What type of license is used?
Usability	Evaluated by examining efforts put in place to ensure that a user, with reasonable technical skills, could use the method	Is there a Wiki, Read the Docs, or GitPage associated with the method? Are there examples of how to use the method?
Reproducibility	Evaluated by examining whether or not the method provides tools or resources to help reproduce the KG construction process and maintain the code base	What tools are provided to help enable reproducibility (e.g., Docker container, Jupyter Notebook, R Markdown)? Does the repository include any form of testing?

Network Statistics. Statistics were calculated to help characterize each build and included counts of nodes, edges, self-loops, minimum, maximum, mean, and median degree, the number of connected components, and the density.

Each OWL-NETS abstracted build was visualized using Gephi [95] (v0.9.2). The OpenOrd Force-Directed layout [96] was applied with an edge cut of 0.5, a fixed time of 0.2, and trained for 750 iterations. To help with interpretation, nodes were colored according to their biological type, which included: anatomical entities, chemical entities, diseases, genomic type, genes, genomic features, organisms, pathways, phenotypes, proteins, sequence features, transcripts, and variants. All KGs were constructed using Docker (v19.03.8) on a Google Cloud Platform N1 Container-Optimized OS instance configured with 24 CPUs, 500 GB of memory, and a 500 GB solid-state drive Boot Disk.

RESULTS

GitHub Open-Source Tool Survey

The search for open-source biomedical KG construction methods on GitHub returned a total of 1,905 repositories. Of these repositories, 231 contained course, tutorial or

presentation material (i.e., manuscript reviews and slide decks), 278 were duplicate or cloned repositories, 79 were KG applications or services, 60 were websites or resource lists, and 1,253 were determined to be irrelevant (i.e., mislabeled, not biomedical, or not a KG construction method). The list of revised methods was supplemented to include 11 additional methods identified through a recent review article [6]. The final list contained the 16 methods (including PheKnowLator), which are shown in [Supplemental Material Table 3](#). Results from applying the five criteria are presented in [Figure 6](#).

For *KG Construction Functionality* (presented in [Supplemental Material Table 4](#)), 81.3% (n=13) of methods provided functionality to download data, 62.5% (n=10) could build multiple types of KGs, 31.3% (n=5) processed free-text, and 37.5% (n=6) processed clinical data. With respect to *Availability* ([Supplemental Material Table 5](#)), 75% (n=12) of the methods were written in Python, 18.8% (n=3) were written in Java-based language, and 6.2% (n=1) were written in R. All of the methods but one were licensed with GPL, 4 MIT, or BSD-3. The *Usability* ([Supplemental Material Table 6](#)) results found that 94.4% (n=17) of the methods provided sample data to help test the method and 80% (n=14) provided literate programming tutorials via R Markdown or Jupyter Notebook. For *Maturity* ([Supplemental Material Table 7](#)), we found that on average, across the 16 methods, the range of commits per year was between 17 and 1,000, over half of the methods included testing and continuous integration, and 43.8% (n=7) provided collaboration guidelines. Finally, with respect to *Reproducibility* ([Supplemental Material Table 8](#)), 75% (n=12) provided tools to enable reproducible workflows and ease installation. Most often, these tools included Docker containers (n=6) and Jupyter or R (n=7) Notebooks. More than 37.5% (n=6) of the methods surveyed used a dependency management program like PyPI or CRAN. Although not an explicit survey question, PheKnowLator was the only method that was able to be verified with a description logic reasoner.

Human Disease Mechanism Knowledge Graph Builds

The knowledge representation used to build the human disease mechanism KGs is shown in [Figure 7](#). Descriptive information for each of the ontologies pre- and post- cleaning is shown in [Table 2](#). As shown in this table, the size of the ontologies varied widely with Chemical Entities of Biological Interest (ChEBI; <https://www.ebi.ac.uk/chebi/>) containing the largest number of edges (n = 5,190,458) and human taxon-specific Protein Ontology (PR; <https://proconsortium.org/>) containing the most classes (n = 148,243). Aside from the RO, which was used only for relations, the Pathway Ontology (PW; <https://rgd.mcg.edu/wg/home/pathway2/>) contained the fewest edges (n = 34,901), and the Sequence Ontology (SO; <http://www.sequenceontology.org/>) contained the fewest classes (n = 2,569). The core set of merged ontologies contained 545,259 classes, 13,748,009 edges, 846 object properties, and 188 individuals. The unique counts of the subjects, objects, relations, and inverse relations for each edge type added to the core set of merged ontologies are shown in [Table 3](#). As shown in this table, the largest edge sets (relations only/inverse relations) added to the KGs consisted of protein-protein (n = 618,069 edges), rna-anatomy (n = 444,668/889,336 edges) relations followed by disease-phenotype (n = 414,193/828,386 edges), and chemical-GOBP (n = 288,921/577,842 edges) relations. The smallest edge sets added to the KG consisted of GOBP-pathway (n = 655 edges) and pathway-GOMF (n = 2,416/4,832 edges), variant-phenotype (n =

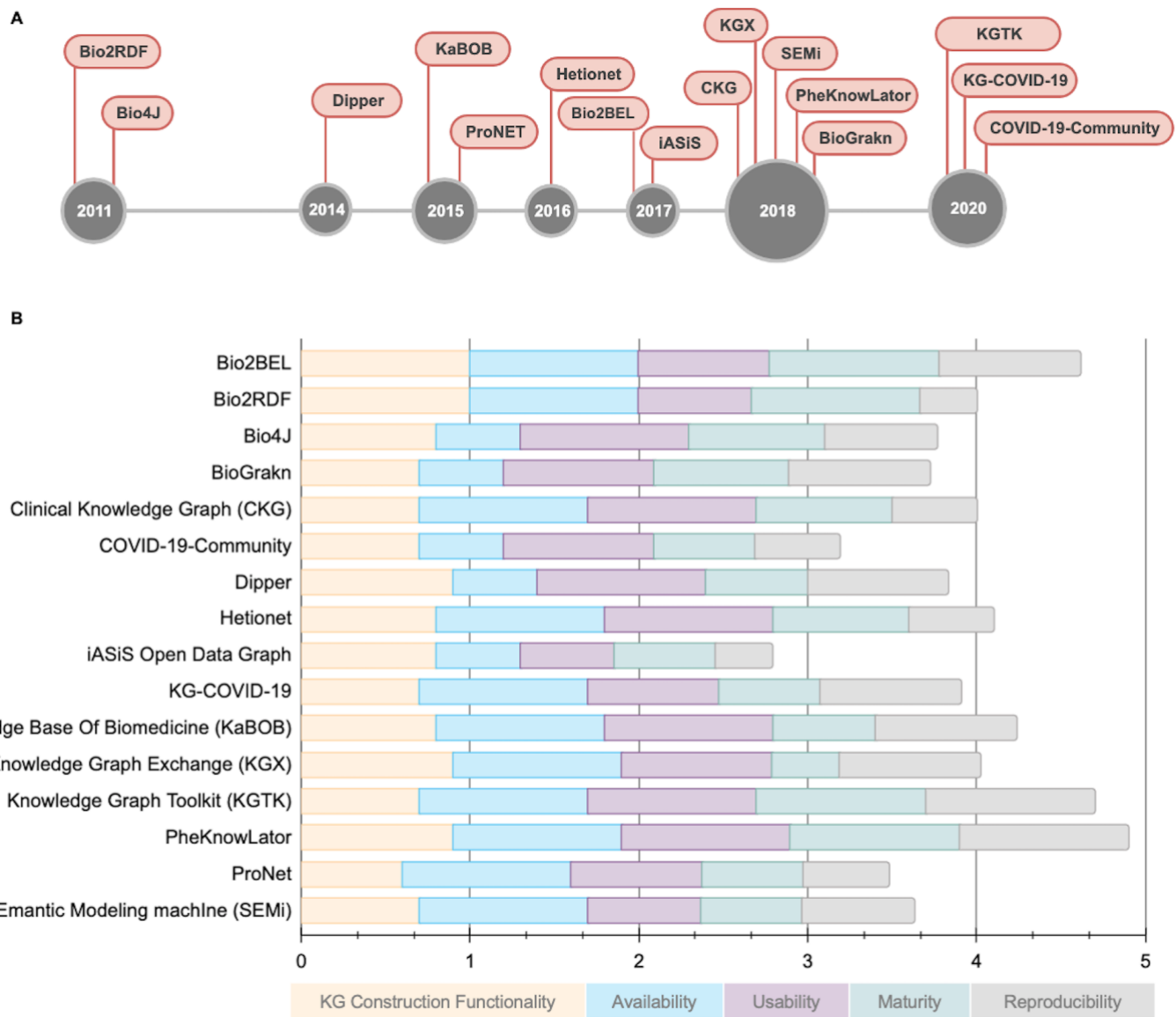


Figure 6. GitHub Survey Results.

Figure 6 (A) Presents the final set of 16 knowledge graph construction methods surveyed from according to the year they were published on GitHub. Figure 5 (B) Presents a chart of the methods evaluated in terms of the different survey categories. As shown in this figure, a total score of 5 points was possible. Acronyms: CKG: Clinical Knowledge Graph; COVID: Coronavirus Disease of 2019; iASiS: Automated Semantic Integration of Disease-Specific Knowledge; KaBOB: Knowledge Base Of Biomedicine; KG: Knowledge Graph; KGX: Knowledge Graph Exchange; KGTK: Knowledge Graph Toolkit; SEMi: SEmantic Modeling machine.

2,526 edges), gene-gene (n = 1,668/3,336 edges), and protein-cofactor (n = 1,999/3,998 edges) relations.

Descriptions of each KG build parameterization are shown in [Table 4](#). As demonstrated by this table, the core set of merged ontologies contained 1,399,756 nodes and 4,044,658 edges. With respect to the full OWL builds, the subclass-based construction approach was the largest (n = 13,803,521 classes and 41,116,791 edges) and both instance-based builds (only relations and inverse relations) were smaller than the subclass-based construction approach KGs with standard relations (n = 8,479,167 nodes and 25,143,729 edges).

All of the full OWL builds, regardless of the construction approach or relation strategy, contained two connected components and three self-loops. All of the builds were highly sparse with the average density across the subclass-based builds ranging from 2.16×10^{-7} - 3.50×10^{-7} and across the instance-based builds ranging from 3.03×10^{-7} - 3.40×10^{-7} . When applying OWL-NETS (visualized in [Figure 8](#)), 449,347 (subclass; [Figure 8 \(A-B\)](#)) to - 235,760 (instance; [Figure 8 \(E-F\)](#)) classes were removed for the standard relation builds and 35,760 (instance; [Figure 8 \(G-H\)](#)) - 587,914 (subclass; [Figure 8 \(C-D\)](#)) classes were removed for the inverse relation builds. For all builds, three miscellaneous classes, 18 `owl:complementOf`, 1,646 `owl:disjointWith`, and 18 classes containing negation (e.g., `pr#lacks part`, `cl#has not completed`) were removed. As there is not yet a reasonable way to represent negation in OWL-NETS without adding nodes that are not biologically meaningful, axioms containing negation were removed. In contrast to the OWL builds, all of the instance-based OWL-NETS builds were larger than the subclass-based builds except for the builds that included standard relations. We also (note that the harmonized instance-based build with standard relations was larger than the harmonized subclass-based build with standard relations). Harmonized instance-based builds had a larger average degree than the harmonized subclass-based builds, which is further confirmed by the visualizations in [Figure 8](#) (9.79 and 12.94 versus 6.68 and 10.26, respectively). It took more than 900 edges to make the subclass-based (relations only: 917; inverse relations: 910) and instance-based (relations only: 917; inverse relations: 943) builds a single connected component. With respect to the harmonized OWL-NETS builds, 161 subclass-based edges required harmonization and 1,162,970 instance-based edges required harmonization.

Computational Performance

All builds were performed on a pre-merged set of cleaned ontologies. When not using the pre-merged set of core ontologies, it took approximately 22 minutes and used an average of 7.29 GiB (1.8-9.8 GiB) of memory to merge all the ontologies. The performance metrics for Steps 1-3 for each KG parameterization are shown in [Figure 9](#). On average, across all KG parameterizations, Step 1 (Data Download) took 3.5 minutes (2.7-5.5 minutes) and used an average of 7.9 GiB of max memory (7.9-8 GiB). Step 2 (Edge List Creation) took an average of 4.8 minutes to complete (4.8-4.9 minutes) and used an average of 39.4 GiB of max memory (38.9-40.4 GiBs). Finally, Step 3 (i.e. Graph Construction) took an average of 391.6 minutes (6.5 hours) to complete (265.9-615.9 minutes or 4.4-10.3 hours) and used an average of 119.7 GiB of max memory (104.3-147.1 GiBs). These plots demonstrate that on average, KG

Table 2. Application of Data Quality Checks to Eleven Open Biomedical Ontologies.

Statistics	CLO	ChEBI	GO	HP	MONDO	PR ^a
Pre-Processed Statistics						
Edges	1,387,096	5,264,571	1,425,434	884,999	2,313,343	2,079,356
Classes	111,712	156,098	62,237	38,843	55,478	148,243
Individuals	41	0	0	0	18	0
Object Properties	116	10	9	231	331	12
Annotation Properties	192	37	53	257	119	11
Connected Components	7	1	2	1	1	3
Data Quality Check Errors						
Value Errors	1	0	0	0	0	0
Identifier Errors	0	0	0	0	0	0
Deprecated Entities	2	18,506	6,430	304	2,305	0
Obsolete Entities	13	0	0	0	0	0
Punning	16	0	0	0	0	0
Consistency ^b	Yes	Yes	Yes	Yes	Yes	Yes
Semantic Heterogeneity	---	---	---	---	---	---
Identifier Alignment	---	---	---	---	---	---
Post-Processed Statistics						
Edges	1,422,153	5,190,485	1,343,218	885,379	2,277,425	2,079,356
Classes	111,696	137,592	55,807	38,530	52,937	148,243
Individuals	33	0	0	0	17	0
Object Properties	112	10	9	231	330	12
Annotation Properties	187	37	53	257	119	11
Connected Components	7	1	2	1	1	3

Statistics	PW	RO	SO	UBERON	VO	Merged ^b
Pre-Processing Statistics						
Edges	35,291	7,970	44,655	752,291	86,454	13,746,883
Classes	2,642	116	2,910	28,738	7,089	548,947
Individuals	0	5	0	0	165	195
Object Properties	1	604	50	242	232	847
Annotation Properties	19	106	41	284	97	656
Connected Components	1	3	1	2	5	8
Data Quality Check Errors						
Value Errors	0	0	0	0	0	0
Identifier Errors	0	0	0	0	2	2
Deprecated Entities	42	11	341	1,570	0	0
Obsolete Entities	0	1	0	0	0	0
Punning	0	0	0	0	0	8
Consistency ^c	Yes	Yes	Yes	Yes	Yes	---
Semantic Heterogeneity	---	---	---	---	---	7
Identifier Alignment	---	---	---	---	---	23,624
Post-Processing Statistics						
Edges	34,901	7,873	41,980	734,768	89,764	13,748,009
Classes	2,600	115	2,569	27,170	7,085	545,259
Individuals	0	5	0	0	165	188
Object Properties	1	594	50	238	232	846
Annotation Properties	19	106	41	284	97	656
Connected Components	1	3	1	2	5	8

Acronyms - CLO: Cell Line Ontology; ChEBI: Chemical Entities of Biological Interest; GO: Gene Ontology; MONDO: Mondo Disease Ontology; PRO: Protein Ontology; PW: Pathway Ontology; RO: Relation Ontology; SO: Sequence Ontology; UBERON: Uber-Anatomy Ontology; VO: Vaccine Ontology.

^aThe PR version references the human (NCBITaxon_9606) subset created for the PheKnowLator ecosystem.

^bConsistency was evaluated using the ELK reasoner. The reasoner was only applied to individual ontologies

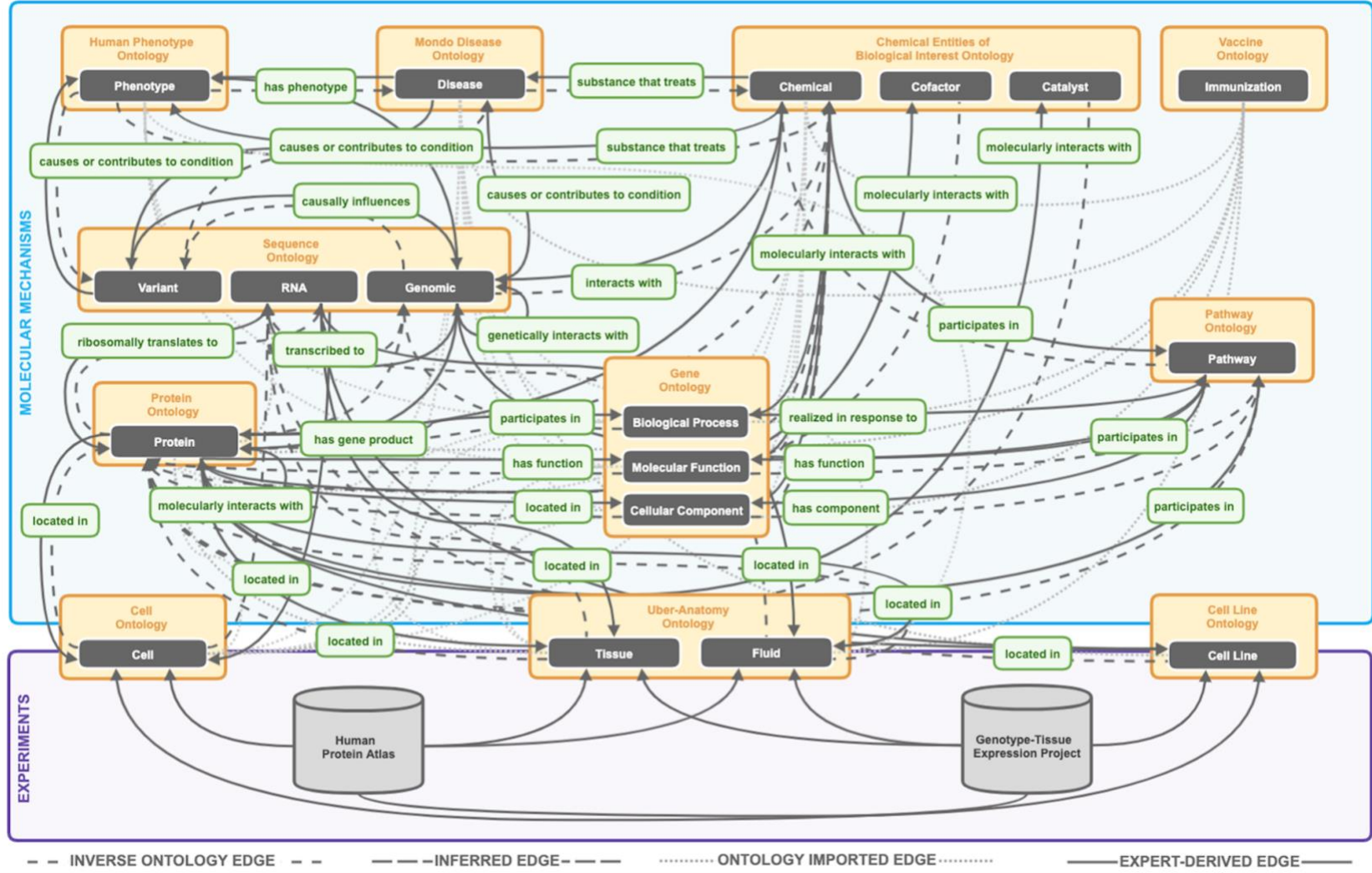


Figure 7. Human Disease Mechanism Knowledge Representation.

The knowledge representation that was used to construct the eight different parameterizations of the PheKnowLator knowledge graphs. The purple box contains experimental data, and the blue box contains molecular mechanisms created by integrating open biomedical ontologies and linked open data.

Table 3. Edge Types Added to the Core Set of Merged Ontologies.

Edge	Relation (Inverse)	Subjects	Objects	Relations	Inverse Relations
chemical-disease	substance that treats (is treated by substance)	4,290	4,574	170,675	341,350
chemical-gene	interacts with*	462	11,981	16,699	33,398
chemical-gobp	molecularly interacts with*	1,338	1,584	288,921	577,842
chemical-gocc	molecularly interacts with*	1,086	250	44,553	89,106
chemical-gomf	molecularly interacts with*	1,105	208	26,165	52,330
chemical-pathway	participates in (has participant)	2,105	2,213	28,691	57,382
chemical-phenotype	substance that treats (is treated by substance)	4,055	1,721	108,452	216,904
chemical-protein	interacts with*	4,179	6,389	65,124	130,248
disease-phenotype	has phenotype (<i>phenotype of</i>)	11,746	9,717	414,193	828,386
gene-disease	causes or contributes to	5,035	4,429	12,735	---
gene-gene	genetically interacts with*	247	263	1,668	3,336
gene-pathway	participates in (has participant)	10,371	1,860	107,025	214,050
gene-phenotype	causes or contributes to	6,785	1,530	23,516	---
gene-protein	has gene product (gene product of)	19,327	19,143	19,534	39,068
gene-rna	transcribed to (transcribed from)	25,529	179,870	182,736	365,472
gobp-pathway	realized in response to	471	665	665	---
pathway-gocc	has component	11,134	99	15,846	---
pathway-gomf	has function (<i>function of</i>)	2,412	726	2,416	4,832
protein-anatomy	located in (<i>location of</i>)	10,747	68	30,682	61,364
protein-catalyst	molecularly interacts with*	3,025	3,734	24,967	49,934
protein-cell	located in (<i>location of</i>)	10,045	128	75,318	150,636
protein-cofactor	molecularly interacts with*	1,585	44	1,999	3,998
protein-gobp	participates in (has participant)	17,527	12,246	137,812	275,624
protein-gocc	located in (<i>location of</i>)	18,427	1,757	81,602	163,204
protein-gomf	has function (<i>function of</i>)	17,779	4,324	68,633	137,266

Edge	Relation (Inverse)	Subjects	Objects	Relations	Inverse Relations
protein-pathway	participates in (has participant)	10,886	2,480	117,585	235,170
protein-protein	molecularly interacts with*	14,230	14,230	618,069	618,069
rna-anatomy	located in (<i>location of</i>)	29,115	103	444,668	889,336
rna-cell	located in (<i>location of</i>)	14,038	130	65,156	130,312
rna-protein	ribosomally translates to (ribosomal translation of)	44,144	19,200	44,147	88,294
variant-disease	causes or contributes to	13,297	3,621	38,129	---
variant-gene	causally influences (causally influenced by)	121,790	3,236	121,790	243,580
variant-phenotype	causes or contributes to	1,824	373	2,526	---

Relations marked with an asterisk (*) are those where symmetric relations were computationally inferred.

Acronyms - gobp: Gene Ontology Biological Process; gocc: Gene Ontology Cellular Component; gomf: Gene Ontology Molecular Function.

Table 4. PheKnowLator Build Descriptive Statistics.

Approach	Relation Strategy	OWL-NETS	Purified	Edges	Nodes	Relations	Connected Components	Self-Loops	Density	Average Degree
Merged Ontologies	NA	No	No	4,044,658	1,399,756	847	2	3	2.06E-06	2.89
Subclass	Relations Only	No	No	25,143,729	8,479,167	847	2	3	3.50E-07	2.97
		Yes	No	4,967,427	743,829	294	1	445	8.98E-06	6.68
			Yes	4,967,429	743,829	293	1	445	8.98E-06	6.68
	Inverse Relations	No	No	41,116,791	13,803,521	847	2	3	2.16E-07	2.98
		Yes	No	7,629,597	743,829	301	1	445	1.38E-05	10.26
			Yes	7,629,599	743,829	300	1	445	1.38E-05	10.26
Instance	Relations Only	No	No	21,770,455	8,479,167	847	2	3	3.03E-07	2.57
		Yes	No	4,967,391	743,829	294	1	409	8.98E-06	6.68
			Yes	7,285,496	743,829	293	1	649	1.32E-05	9.79
	Inverse Relations	No	No	24,432,633	8,479,167	847	2	3	3.40E-07	2.88
		Yes	No	7,629,594	743,829	301	1	409	1.38E-05	10.26
			Yes	9,624,232	743,829	300	1	650	1.74E-05	12.94

parameterizations constructed using the subclass-based construction approach took roughly the same amount of time and used roughly the same maximum amount of memory as the instance-based construction approach. Additionally, regardless of the construction approach, KGs that included inverse relations and performed OWL decoding took longer to run and used more memory, on average.

DISCUSSION

This work introduces PheKnowLator, the first non-commercial open source fully customizable KG construction framework that enables users to build complex KGs that are Semantic Web-compliant and amenable to automatic OWL reasoning, conform to contemporary property graph standards, and are importable by today's popular graph toolkits. PheKnowLator provides this functionality by offering multiple build types, automatically including inverse edges, creating OWL-decoded KGs to support automated deductive reasoning, and outputting KGs in several formats (e.g., edge lists, OWL API-formatted RDF/XML, and graph-pickled NetworkX MultiDiGraphs). By providing flexibility in the way relations are modeled and facilitating the creation of OWL-NETS property graphs, PheKnowLator also enables the use of cutting-edge graph-based learning and sophisticated inference algorithms.

Surveying open-source tools on GitHub for constructing KGs identified 14 similar methods. Using five criteria, these methods were compared to PheKnowLator, revealing that while PheKnowLator has comparable functionality for constructing KGs, most methods were built with specific use cases in mind, whereas PheKnowLator was built to help users solve a wide variety of challenges (some of which are mentioned/described below). This survey also identified areas in which PheKnowLator could be improved, specifically, with respect to the types of input data it can process; when compared to the other methods, three methods (i.e., iASiS Open Data Graph, BioGrakn, and Bio2RDF) were found to process free-text and clinical data, which PheKnowLator does not currently do. PheKnowLator is grounded in ontologies and at the time of the survey, was the only method that could enable description logic-based reasoning. With respect to *Availability* and *Usability*, PheKnowLator and six other methods (i.e., the Clinical Knowledge Graph, Hetionet, KaBOB, KG-COVID-19, KGTK, and the Knowledge Graph Exchange) are equally open source, provide useful examples, and well documented. In terms of *Maturity*, PheKnowLator was one of only three methods (i.e., KG-COVID-19 and KGTK) to provide collaboration documentation, include testing and continuous integration, and actively engage with a user base via GitHub Issue trackers. Similarly, for *Reproducibility*, PheKnowLator was found to be one of the best methods for providing tools and resources that enable fully reproducible builds.

When leveraging PheKnowLator to build KGs representing human disease mechanisms, descriptive statistics and visual inspection confirmed that each of the 12 builds resulted in KGs with different topologies. As expected, the OWL builds were significantly larger than the OWL-NETS property graphs, and instance-based builds were larger for all builds except for the subclass-based standard relation OWL-NETS builds. This finding is interesting considering that the computational performance of the different builds was roughly the same.

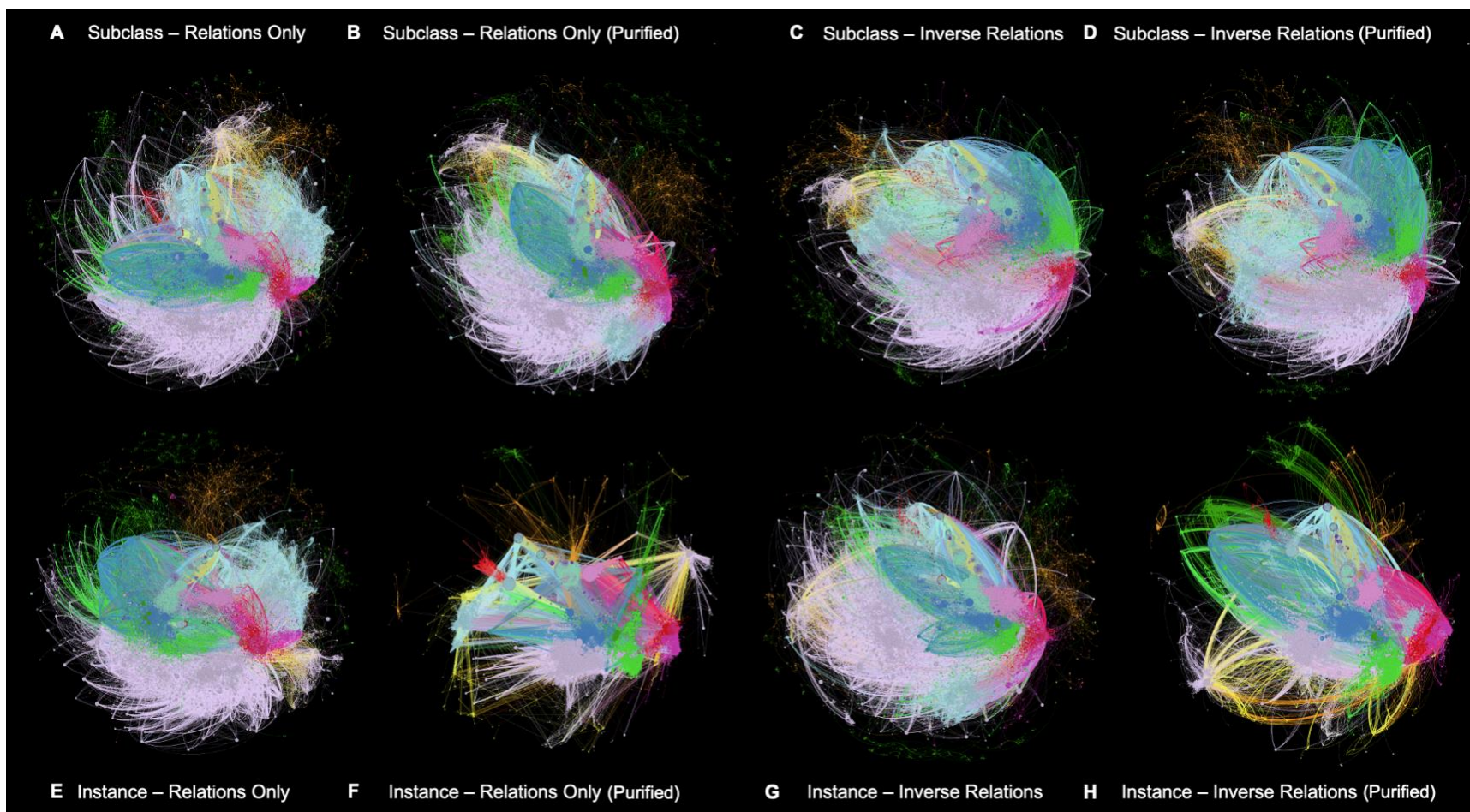


Figure 8. Visualizing PheKnowLator OWL-NETS Builds.

The figure visualizes eight different OWL-NETS builds: (A) subclass-based with standard relations; (B) subclass-based with standard relations (purified); (C) subclass-based with inverse relations; (D) subclass-based with inverse relations (purified); (E) instance-based with standard relations; (F) instance-based with standard relations (purified); (G) instance-based with inverse relations; (H) instance-based with inverse relations (purified). Nodes are colored by entity type: anatomical entities (light blue), chemical entities (light purple), diseases (red), genes (purple), genomic features (light green), organisms (yellow), pathways (dark green), phenotypes (magenta), proteins (dark blue), sequence features (orange), transcripts (turquoise), and variants (light pink). This figure was created using the OpenOrd Force-Directed layout [93] provided by Gephi [92] (v0.9.2).

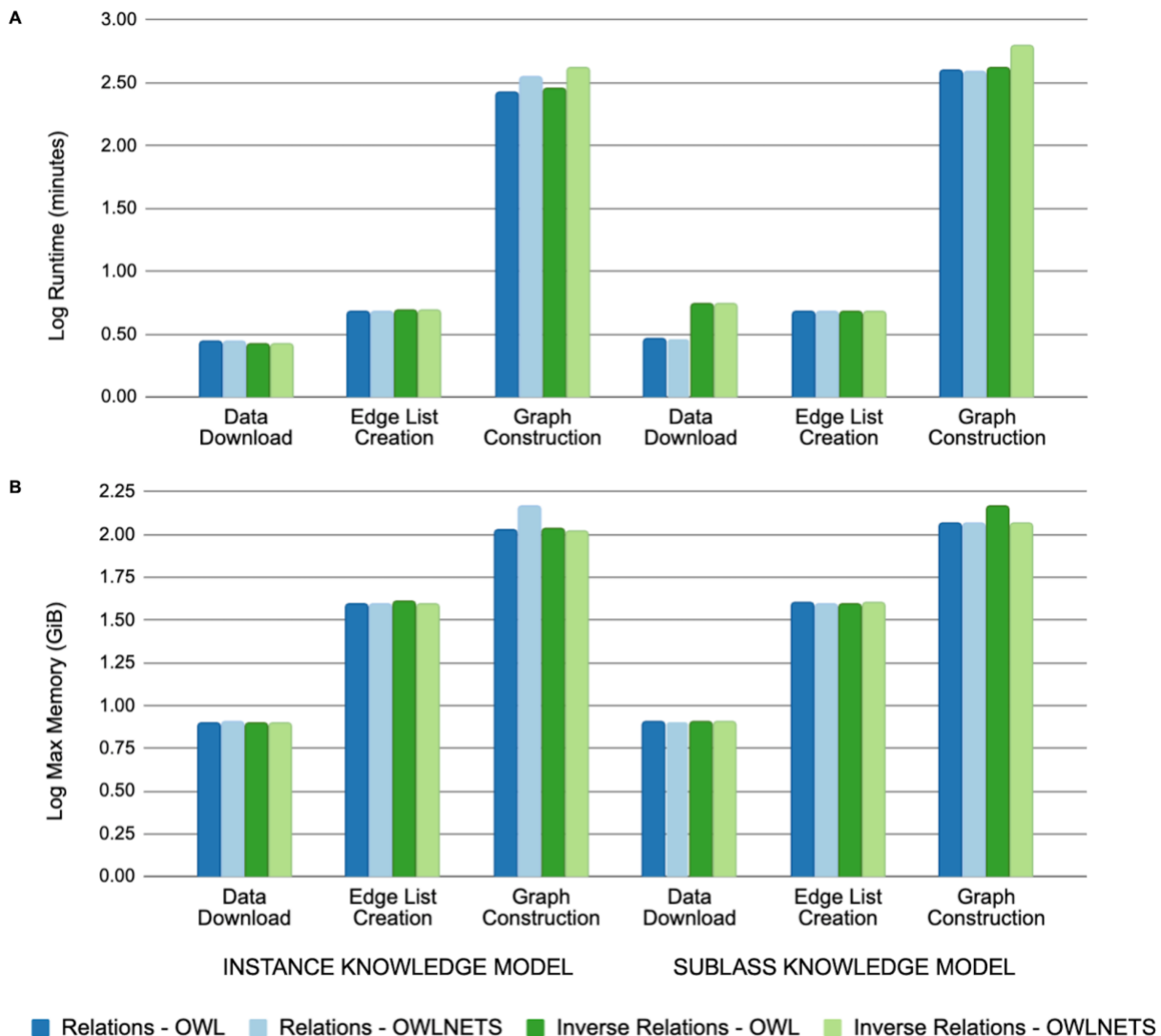


Figure 9. PheKnowLator Computational Performance.

The computational performance for each build step by build configuration with respect to the (A) log runtime and (B) log max memory use (GiB) is presented.

Limitations and Future Work

This work has limitations. Computational performance metrics were only computed over a single build run due to the amount of resources required to build the KGs. While it is not expected that the results for these metrics would significantly change, small deviations related to the Google Cloud Platform or constraints on data providers with respect to accessing build data, could result in different outcomes. Currently, the PheKnowLator Ecosystem relies heavily on OWLTools, but newer Semantic Web tools like ROBOT [97] exist and should be used because it allows for the integration of the OWL API and has improved Jena-based functionality. Validating very large KGs like the ones produced by PheKnowLator is challenging but important.

Additional validation of the work presented in this section is needed to demonstrate the usefulness of these KGs and provide more insight into which builds are best for which types of downstream applications.

Current Collaborations and Avenues for Future Work

The PheKnowLator Ecosystem has already fostered many exciting collaborations and is currently being used by the following projects:

- PheKnowLator KGs have recently been used in applications of toxicogenomic mechanistic inference [98] and biomedical hypergraphs [99].
- Although results are not yet available, PheKnowLator is currently included in the Continuous Evaluation of Relational Learning in Biomedicine (CERLIB; <https://biochallenge.bio2vec.net/>) task. This task aims to provide a means for evaluating prediction models as new knowledge becomes available over time. Results from this task will provide insight into the usefulness of the PheKnowLator builds and will be used to identify areas where the Ecosystem can be improved.
- PheKnowLator has been deployed to create a disease-specific KG that combines ontology-grounded resources with literature-derived computable knowledge from machine reading that fills in knowledge gaps and and resolves machine-reading errors in the extracted knowledge [100]. The resulting KG is then searched to identify confounders with the goal of improving causal inference from observational clinical data by addressing confounding bias.
- The PheKnowLator ecosystem is being used in another project to generate hypotheses for potential pharmacokinetic natural product-drug interactions. The project is creating a KG involving biomedical ontologies, natural product-ontology extensions, and machine reading from literature (<https://github.com/sanyabt/napdi-kg>). Preliminary results of this work were presented at the 2021 International Society of Computational Biology [101] and the American Medical Informatics Association Informatics Summit 2022.
- Currently we are working with PheKnowLator KGs to analyze variant-disease associations using cutting edge graph embedding AI tools [102], and we plan to apply random walk- based embedding methods for heterogeneous graphs to analyze the full builds of the PheKnowLator KGs representing human disease mechanisms [103].
- The NIH Common Fund Human BioMolecular Atlas Program (HuBMAP)[104] needed to assemble a KG based on its own preferred graph schema[105–107], with one focus being to maximize leverage of external references among ontologies for translation. The initial effort used the Unified Medical Language System (UMLS) [108] for its data and model construction. The need to add 12 OWL-formatted ontologies to the graph, thereby interoperating the “UMLS-world” directly with the “OWL-world” (e.g., integration of Uberon and Cell Ontology with Foundation Model of Anatomy, etc...). This key problem in the community was solved by using OWL-NETS [84] to extract the relevant assertions from the OWL files into tables followed by a transformation and load script (also in

Python) into the HuBMAP KG neo4j import format developed for UMLS ingestion. The OWL-NETS node metadata and edge list tables are now being used as a standard to implement ingestion of other operational ontologies (whether in OWL or not) into HuBMAP.

We would like to invite the community to collaborate with us to examine the utility of PheKnowLator across an even wider variety of use cases.

Avenues for Future Work

Evaluation

Although we have generated embeddings for earlier versions of PheKnowLator, the pipelines to create embeddings for the current builds are still under construction. Future work will leverage these embeddings to provide additional validation of the PheKnowLator Ecosystem and subsequent KGs.

User Experience and Usability

While we are working hard to improve the usability of PheKnowLator, this is an area that we will continue to build on in the future. Aside from improving documentation and working on scripts...

- Improving scripts which automate the preparation of build docs
- Having a workshop
- Adding YouTube videos to help introduce users to the codebase, tools, and functionality

CONCLUSION

PheKnowLator is an ecosystem for the FAIR construction of ontologically grounded KGs. PheKnowLator KGs can be built under alternative knowledge models, using unidirectional or bidirectional relations, and with or without semantic property graph abstraction. Although additional experiments are needed to demonstrate the value of the different KGs that can be produced by this Ecosystem, PheKnowLator is one of the first fully customizable open-source KG construction frameworks able to provide a wide range of functionality without compromising usability.

REFERENCES

1. Agrawal R, Prabakaran S. Big data in digital healthcare: lessons learnt and recommendations for general practice. *Heredity* [Internet]. 2020 Apr;124(4):525–34. Available from: <http://dx.doi.org/10.1038/s41437-020-0303-2>
2. Bietz MJ, Bloss CS, Calvert S, Godino JG, Gregory J, Claffey MP, et al. Opportunities and challenges in the use of personal health data for health research. *J Am Med Inform Assoc* [Internet]. 2016 Apr;23(e1):e42–8. Available from: <http://dx.doi.org/10.1093/jamia/ocv118>
3. Hulsén T, Jamuar SS, Moody AR, Karnes JH, Varga O, Hedensted S, et al. From Big Data to Precision Medicine. *Front Med* [Internet]. 2019 Mar 1;6:34. Available from: <http://dx.doi.org/10.3389/fmed.2019.00034>
4. Fröhlich H, Balling R, Beerenwinkel N, Kohlbacher O, Kumar S, Lengauer T, et al. From hype to reality: data science enabling personalized medicine. *BMC Med* [Internet]. 2018 Aug 27;16(1):150. Available from: <http://dx.doi.org/10.1186/s12916-018-1122-7>
5. Livingston KM, Bada M, Baumgartner WA Jr, Hunter LE. KaBOB: ontology-based semantic integration of biomedical databases. *BMC Bioinformatics* [Internet]. 2015 Apr 23;16:126. Available from: <http://dx.doi.org/10.1186/s12859-015-0559-3>
6. Callahan TJ, Tripodi IJ, Pielke-Lombardo H, Hunter LE. Knowledge-Based Biomedical Data Science. *Annu Rev Biomed Data Sci* [Internet]. 2020 Jul 20; Available from: <https://doi.org/10.1146/annurev-biodatasci-010820-091627>
7. Vidal M, Cusick ME, Barabási A-L. Interactome networks and human disease. *Cell* [Internet]. 2011 Mar 18;144(6):986–98. Available from: <http://dx.doi.org/10.1016/j.cell.2011.02.016>
8. Yi H-C, You Z-H, Huang D-S, Kwok CK. Graph representation learning in bioinformatics: trends, methods and applications. *Brief Bioinform* [Internet]. 2021 Sep 1; Available from: <http://dx.doi.org/10.1093/bib/bbab340>
9. Hamilton WL, Ying R, Leskovec J. Representation Learning on Graphs: Methods and Applications [Internet]. *arXiv [cs.SI]*. 2017. Available from: <http://arxiv.org/abs/1709.05584>
10. Nicholson DN, Greene CS. Constructing knowledge graphs and their biomedical applications. *Comput Struct Biotechnol J* [Internet]. 2020 Jun 2;18:1414–28. Available from: <http://dx.doi.org/10.1016/j.csbj.2020.05.017>
11. Wilkinson MD, Dumontier M, Aalbersberg IJJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* [Internet]. 2016 Mar 15;3:160018. Available from: <http://dx.doi.org/10.1038/sdata.2016.18>
12. Ehrlinger L, Wöß W. Towards a Definition of Knowledge Graphs. *SEMANTiCS (Posters, Demos, SuCCESS)* [Internet]. 2016;48. Available from: https://www.researchgate.net/profile/Wolfram_Woess/publication/323316736_Towards_a_Definition_of_Knowledge_Graphs/links/5a8d6e8f0f7e9b27c5b4b1c3/Towards-a-Definition-of-Knowledge-Graphs.pdf
13. Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutierrez C, et al. Knowledge

- Graphs [Internet]. arXiv [cs.AI]. 2020. Available from: <http://arxiv.org/abs/2003.02320>
14. Ji S, Pan S, Cambria E, Marttinen P, Yu PS. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Trans Neural Netw Learn Syst* [Internet]. 2021 Apr 26;PP. Available from: <http://dx.doi.org/10.1109/TNNLS.2021.3070843>
 15. OWL Web Ontology Language Overview [Internet]. [cited 2019 Oct 1]. Available from: <https://www.w3.org/TR/owl-features/>
 16. Krötzsch M, Simancik F, Horrocks I. A Description Logic Primer [Internet]. arXiv [cs.AI]. 2012. Available from: <http://arxiv.org/abs/1201.4089>
 17. Miller E. An introduction to the resource description framework. *Bull Am Soc Inf Sci* [Internet]. 2005 Jan 31;25(1):15–9. Available from: <https://onlinelibrary.wiley.com/doi/10.1002/bult.105>
 18. Reese JT, Unni D, Callahan TJ, Cappelletti L, Ravanmehr V, Carbon S, et al. KG-COVID-19: A Framework to Produce Customized Knowledge Graphs for COVID-19 Response. *Patterns (N Y)* [Internet]. 2021 Jan 8;2(1):100155. Available from: <http://dx.doi.org/10.1016/j.patter.2020.100155>
 19. Piñero J, Bravo À, Queralt-Rosinach N, Gutiérrez-Sacristán A, Deu-Pons J, Centeno E, et al. DisGeNET: a comprehensive platform integrating information on human disease-associated genes and variants [Internet]. Vol. 45, *Nucleic Acids Research*. 2017. p. D833–9. Available from: <http://dx.doi.org/10.1093/nar/gkw943>
 20. Breit A, Ott S, Agibetov A, Samwald M. OpenBioLink: a benchmarking framework for large-scale biomedical link prediction. *Bioinformatics* [Internet]. 2020 Jul 1;36(13):4097–8. Available from: <http://dx.doi.org/10.1093/bioinformatics/btaa274>
 21. Belleau F, Nolin M-A, Tourigny N, Rigault P, Morissette J. Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *J Biomed Inform* [Internet]. 2008 Oct;41(5):706–16. Available from: <http://dx.doi.org/10.1016/j.jbi.2008.03.004>
 22. Himmelstein DS, Lizee A, Hessler C, Brueggeman L, Chen SL, Hadley D, et al. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife* [Internet]. 2017 Sep 22;6. Available from: <http://dx.doi.org/10.7554/eLife.26726>
 23. Home [Internet]. [cited 2021 Nov 20]. Available from: <https://spoke.ucsf.edu/>
 24. Kilicoglu H, Shin D, Fiszman M, Rosemblat G, Rindflesch TC. SemMedDB: a PubMed-scale repository of biomedical semantic predications. *Bioinformatics* [Internet]. 2012 Dec 1;28(23):3158–60. Available from: <http://dx.doi.org/10.1093/bioinformatics/bts591>
 25. Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol* [Internet]. 2007 Nov;25(11):1251–5. Available from: <http://dx.doi.org/10.1038/nbt1346>
 26. Jackson RC, Matentzoglou N, Overton JA, Vita R, Balhoff JP, Buttigieg PL, et al. OBO Foundry in 2021: Operationalizing Open Data Principles to Evaluate Ontologies [Internet]. *bioRxiv*. 2021 [cited 2021 Jul 14]. p. 2021.06.01.446587. Available from: <https://www.biorxiv.org/content/biorxiv/early/2021/06/02/2021.06.01.446587>

27. Köhler S, Gargano M, Matentzoglou N, Carmody LC, Lewis-Smith D, Vasilevsky NA, et al. The Human Phenotype Ontology in 2021. *Nucleic Acids Res* [Internet]. 2021 Jan 8;49(D1):D1207–17. Available from: <http://dx.doi.org/10.1093/nar/gkaa1043>
28. Disease ontology - institute for genome sciences @ university of Maryland [Internet]. [cited 2021 Nov 22]. Available from: <https://disease-ontology.org/>
29. Protein Ontology [Internet]. [cited 2021 Nov 22]. Available from: <https://proconsortium.org/>
30. Thomas PD, Hill DP, Mi H, Osumi-Sutherland D, Van Auken K, Carbon S, et al. Gene Ontology Causal Activity Modeling (GO-CAM) moves beyond GO annotations to structured descriptions of biological functions and systems. *Nat Genet* [Internet]. 2019 Sep 23; Available from: <http://dx.doi.org/10.1038/s41588-019-0500-1>
31. Gene Ontology Resource [Internet]. [cited 2021 Nov 22]. Available from: <http://geneontology.org/>
32. Knowledge Graph Construction Techniques. *Journal of Computer Research and Development* [Internet]. 2016 Mar; Available from: http://en.cnki.com.cn/Article_en/CJFDTotal-JFYZ201603009.htm
33. Bunescu R, Ge R, Kate RJ, Marcotte EM, Mooney RJ, Ramani AK, et al. Comparative experiments on learning information extractors for proteins and their interactions. *Artif Intell Med* [Internet]. 2005 Feb;33(2):139–55. Available from: <http://dx.doi.org/10.1016/j.artmed.2004.07.016>
34. Pysalo S, Ginter F, Heimonen J, Björne J, Boberg J, Järvinen J, et al. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics* [Internet]. 2007 Feb 9;8:50. Available from: <http://dx.doi.org/10.1186/1471-2105-8-50>
35. Ding J, Berleant D, Nettleton D, Wurtele E. Mining MEDLINE: abstracts, sentences, or phrases? *Pac Symp Biocomput* [Internet]. 2002;326–37. Available from: http://dx.doi.org/10.1142/9789812799623_0031
36. Fundel K, Küffner R, Zimmer R. RelEx--relation extraction using dependency parse trees. *Bioinformatics* [Internet]. 2007 Feb 1;23(3):365–71. Available from: <http://dx.doi.org/10.1093/bioinformatics/btl616>
37. van Mulligen EM, Fourrier-Reglat A, Gurwitz D, Molokhia M, Nieto A, Trifiro G, et al. The EU-ADR corpus: annotated drugs, diseases, targets, and their relationships. *J Biomed Inform* [Internet]. 2012 Oct;45(5):879–84. Available from: <http://dx.doi.org/10.1016/j.jbi.2012.04.004>
38. Bravo À, Piñero J, Queralt-Rosinach N, Rautschka M, Furlong LI. Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. *BMC Bioinformatics* [Internet]. 2015 Feb 21;16:55. Available from: <http://dx.doi.org/10.1186/s12859-015-0472-9>
39. Lee H-J, Shim S-H, Song M-R, Lee H, Park JC. CoMAGC: a corpus with multi-faceted annotations of gene-cancer relations. *BMC Bioinformatics* [Internet]. 2013 Nov 14;14:323. Available from: <http://dx.doi.org/10.1186/1471-2105-14-323>

40. Bada M, Eckert M, Evans D, Garcia K, Shipley K, Sitnikov D, et al. Concept annotation in the CRAFT corpus. *BMC Bioinformatics* [Internet]. 2012 Jul 9;13:161. Available from: <http://dx.doi.org/10.1186/1471-2105-13-161>
41. Ye Q, Hsieh C-Y, Yang Z, Kang Y, Chen J, Cao D, et al. A unified drug-target interaction prediction framework based on knowledge graph and recommendation system. *Nat Commun* [Internet]. 2021 Nov 22;12(1):6775. Available from: <http://dx.doi.org/10.1038/s41467-021-27137-3>
42. Mohamed SK, Nováček V, Nounu A. Discovering protein drug targets using knowledge graph embeddings. *Bioinformatics* [Internet]. 2020 Jan 15;36(2):603–10. Available from: <http://dx.doi.org/10.1093/bioinformatics/btz600>
43. Al-Saleem J, Granet R, Ramakrishnan S, Ciancetta NA, Saveson C, Gessner C, et al. Knowledge Graph-Based Approaches to Drug Repurposing for COVID-19. *J Chem Inf Model* [Internet]. 2021 Aug 23;61(8):4058–67. Available from: <http://dx.doi.org/10.1021/acs.jcim.1c00642>
44. Zhu Y, Che C, Jin B, Zhang N, Su C, Wang F. Knowledge-driven drug repurposing using a comprehensive drug knowledge graph. *Health Informatics J* [Internet]. 2020 Dec;26(4):2737–50. Available from: <http://dx.doi.org/10.1177/1460458220937101>
45. Sosa DN, Derry A, Guo M, Wei E, Brinton C, Altman RB. A Literature-Based Knowledge Graph Embedding Method for Identifying Drug Repurposing Opportunities in Rare Diseases. *Pac Symp Biocomput* [Internet]. 2020;25:463–74. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/31797619>
46. Zhang X, Che C. Drug Repurposing for Parkinson's Disease by Integrating Knowledge Graph Completion Model and Knowledge Fusion of Medical Literature. *Future Internet* [Internet]. 2021 Jan 8 [cited 2021 Dec 7];13(1):14. Available from: <https://www.mdpi.com/1999-5903/13/1/14>
47. Du J, Li X. A Knowledge Graph of Combined Drug Therapies Using Semantic Predications From Biomedical Literature: Algorithm Development. *JMIR Med Inform* [Internet]. 2020 Apr 28;8(4):e18323. Available from: <http://dx.doi.org/10.2196/18323>
48. Rak R, Rowley A, Black W, Ananiadou S. Argo: an integrative, interactive, text mining-based workbench supporting curation. *Database* [Internet]. 2012 Mar 20;2012:bas010. Available from: <http://dx.doi.org/10.1093/database/bas010>
49. Kwon D, Kim S, Shin S-Y, Chatr-aryamontri A, Wilbur WJ. Assisting manual literature curation for protein-protein interactions using BioQRator. *Database* [Internet]. 2014 Jul 22;2014. Available from: <http://dx.doi.org/10.1093/database/bau067>
50. Kim J-D, Cohen KB, Kim J-J. PubAnnotation-query: a search tool for corpora with multi-layers of annotation. *BMC Proc* [Internet]. 2015 Aug 6;9(5):A3. Available from: <https://doi.org/10.1186/1753-6561-9-S5-A3>
51. Müller H-M, Van Auken KM, Li Y, Sternberg PW. Textpresso Central: a customizable platform for searching, text mining, viewing, and curating biomedical literature. *BMC Bioinformatics* [Internet]. 2018 Mar 9;19(1):94. Available from: <http://dx.doi.org/10.1186/s12859-018-2103-8>

52. Loster M, Naumann F, Ehmüller J, Feldmann B. CurEx: A System for Extracting, Curating, and Exploring Domain-Specific Knowledge Graphs from Text. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management [Internet]. New York, NY, USA: Association for Computing Machinery; 2018 [cited 2021 Nov 22]. p. 1883–6. (CIKM '18). Available from: <https://doi.org/10.1145/3269206.3269229>
53. Cejuela JM, Vinchurkar S, Goldberg T, Prabhu Shankar MS, Baghudana A, Bojchevski A, et al. LocText: relation extraction of protein localizations to assist database curation. *BMC Bioinformatics* [Internet]. 2018 Jan 17;19(1):15. Available from: <http://dx.doi.org/10.1186/s12859-018-2021-9>
54. Junge A, Jensen LJ. CoCoScore: context-aware co-occurrence scoring for text mining applications using distant supervision. *Bioinformatics* [Internet]. 2020 Jan 1;36(1):264–71. Available from: <http://dx.doi.org/10.1093/bioinformatics/btz490>
55. Rastegar-Mojarad M, Elayavilli RK, Li D, Prasad R, Liu H. A new method for prioritizing drug repositioning candidates extracted by literature-based discovery. In: 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) [Internet]. 2015. p. 669–74. Available from: <http://dx.doi.org/10.1109/BIBM.2015.7359766>
56. Frijters R, van Vugt M, Smeets R, van Schaik R, de Vlieg J, Alkema W. Literature mining for the discovery of hidden connections between drugs, genes and diseases. *PLoS Comput Biol* [Internet]. 2010 Sep 23;6(9). Available from: <http://dx.doi.org/10.1371/journal.pcbi.1000943>
57. Westergaard D, Stærfeldt H-H, Tønsberg C, Jensen LJ, Brunak S. A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts. *PLoS Comput Biol* [Internet]. 2018 Feb;14(2):e1005962. Available from: <http://dx.doi.org/10.1371/journal.pcbi.1005962>
58. Pletscher-Frankild S, Pallejà A, Tsafou K, Binder JX, Jensen LJ. DISEASES: text mining and data integration of disease-gene associations. *Methods* [Internet]. 2015 Mar;74:83–9. Available from: <http://dx.doi.org/10.1016/j.ymeth.2014.11.020>
59. Szklarczyk D, Franceschini A, Wyder S, Forslund K, Heller D, Huerta-Cepas J, et al. STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res* [Internet]. 2015 Jan;43(Database issue):D447–52. Available from: <http://dx.doi.org/10.1093/nar/gku1003>
60. Singhal A, Simmons M, Lu Z. Text Mining Genotype-Phenotype Relationships from Biomedical Literature for Database Curation and Precision Medicine. *PLoS Comput Biol* [Internet]. 2016 Nov;12(11):e1005017. Available from: <http://dx.doi.org/10.1371/journal.pcbi.1005017>
61. Minervini P, d'Amato C, Fanizzi N, Esposito F. Leveraging the schema in latent factor models for knowledge graph completion. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing [Internet]. New York, NY, USA: Association for Computing Machinery; 2016 [cited 2021 Nov 22]. p. 327–32. (SAC '16). Available from: <https://doi.org/10.1145/2851613.2851841>
62. Chen L, Cui J, Tang X, Qian Y, Li Y, Zhang Y. RLPPath: a knowledge graph link prediction method using reinforcement learning based attentive relation path searching and

- representation learning. *Applied Intelligence* [Internet]. 2021 Jul 27; Available from: <https://doi.org/10.1007/s10489-021-02672-0>
63. Wang M, Qiu L, Wang X. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* [Internet]. 2021 Mar 16 [cited 2021 Nov 22];13(3):485. Available from: <https://www.mdpi.com/1036316>
 64. Percha B, Altman RB. A global network of biomedical relationships derived from text. *Bioinformatics* [Internet]. 2018 Aug 1;34(15):2614–24. Available from: <http://dx.doi.org/10.1093/bioinformatics/bty114>
 65. Zhang Q, Sun Z, Hu W, Chen M, Guo L, Qu Y. Multi-view Knowledge Graph Embedding for Entity Alignment [Internet]. *arXiv [cs.AI]*. 2019. Available from: <http://arxiv.org/abs/1906.02390>
 66. Collarana D, Galkin M, Traverso-Ribón I, Lange C, Vidal M-E, Auer S. Semantic Data Integration for Knowledge Graph Construction at Query Time. In: 2017 IEEE 11th International Conference on Semantic Computing (ICSC) [Internet]. 2017. p. 109–16. Available from: <http://dx.doi.org/10.1109/ICSC.2017.85>
 67. Kertkeidkachorn N, Ichise R. T2kg: An end-to-end system for creating knowledge graph from unstructured text. In: Workshops at the Thirty-First AAAI Conference on Artificial Intelligence [Internet]. 2017. Available from: <https://www.aaai.org/ocs/index.php/WS/AAAIW17/paper/viewPaper/15129>
 68. Shi B, Weninger T. Open-World Knowledge Graph Completion. In: Thirty-Second AAAI Conference on Artificial Intelligence [Internet]. 2018 [cited 2021 Nov 22]. Available from: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/16055>
 69. Ebeid IA, Hassan M, Wanyan T, Roper J, Seal A, Ding Y. Biomedical Knowledge Graph Refinement and Completion Using Graph Representation Learning and Top-K Similarity Measure. In: Diversity, Divergence, Dialogue [Internet]. Springer International Publishing; 2021. p. 112–23. Available from: http://dx.doi.org/10.1007/978-3-030-71292-1_10
 70. Deng Y, Li Y, Shen Y, Du N, Fan W, Yang M, et al. MedTruth: A Semi-supervised Approach to Discovering Knowledge Condition Information from Multi-Source Medical Data [Internet]. *arXiv [cs.DB]*. 2018. Available from: <http://arxiv.org/abs/1809.10404>
 71. Bosselut A, Rashkin H, Sap M, Malaviya C, Celikyilmaz A, Choi Y. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction [Internet]. *arXiv [cs.CL]*. 2019. Available from: <http://arxiv.org/abs/1906.05317>
 72. Wright D. NormCo: Deep Disease Normalization for Biomedical Knowledge Base Construction [Internet]. UC San Diego; 2019 [cited 2019 Sep 24]. Available from: <https://escholarship.org/uc/item/3410q7zk>
 73. The Challenge [Internet]. [cited 2021 Nov 22]. Available from: <http://bioasq.org/>
 74. Sáez C, Martínez-Miranda J, Robles M, García-Gómez JM. Organizing data quality assessment of shifting biomedical data. In: Quality of Life through Quality of Information [Internet]. IOS Press; 2012. p. 721–5. Available from: <https://ebooks.iospress.nl/doi/10.3233/978-1-61499-101-4-721>

75. Hoffman S, Podgurski A. The use and misuse of biomedical data: is bigger really better? *Am J Law Med* [Internet]. 2013;39(4):497–538. Available from: <http://dx.doi.org/10.1177/009885881303900401>
76. Hu W, Zaveri A, Qiu H, Dumontier M. Cleaning by clustering: methodology for addressing data quality issues in biomedical metadata. *BMC Bioinformatics* [Internet]. 2017 Sep 18;18(1):415. Available from: <http://dx.doi.org/10.1186/s12859-017-1832-4>
77. Amith M, He Z, Bian J, Lossio-Ventura JA, Tao C. Assessing the practice of biomedical ontology evaluation: Gaps and opportunities. *J Biomed Inform* [Internet]. 2018 Apr;80:1–13. Available from: <http://dx.doi.org/10.1016/j.jbi.2018.02.010>
78. Zhu X, Fan J-W, Baorto DM, Weng C, Cimino JJ. A review of auditing methods applied to the content of controlled biomedical terminologies. *J Biomed Inform* [Internet]. 2009 Jun;42(3):413–25. Available from: <http://dx.doi.org/10.1016/j.jbi.2009.03.003>
79. Davis R, Shrobe H, Szolovits P. What is a knowledge representation? *AI magazine* [Internet]. 1993;14(1):17–17. Available from: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/1029>
80. Hunter LE. Knowledge-based biomedical Data Science. *EPJ Data Sci* [Internet]. 2017 Dec 8;1(1-2):19–25. Available from: <http://dx.doi.org/10.3233/DS-170001>
81. Demir E, Cary MP, Paley S, Fukuda K, Lemer C, Vastrik I, et al. The BioPAX community standard for pathway data sharing. *Nat Biotechnol* [Internet]. 2010 Sep;28(9):935–42. Available from: <http://dx.doi.org/10.1038/nbt.1666>
82. Biological Expression Language [Internet]. [cited 2021 Nov 23]. Available from: <https://biological-expression-language.github.io/>
83. Lam HYK, Marenco L, Shepherd GM, Miller PL, Cheung K-H. Using web ontology language to integrate heterogeneous databases in the neurosciences. *AMIA Annu Symp Proc* [Internet]. 2006;464–8. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/17238384>
84. Callahan TJ, Baumgartner WA, Bada M, Stefanski AL, Tripodi I, White EK, et al. OWL-NETS: Transforming OWL Representations for Improved Network Inference. In: *Biocomputing 2018* [Internet]. WORLD SCIENTIFIC; 2017. p. 133–44. Available from: https://doi.org/10.1142/9789813235533_0013
85. Rebele T, Suchanek F, Hoffart J, Biega J, Kuzey E, Weikum G. YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames. In: *The Semantic Web – ISWC 2016* [Internet]. Springer International Publishing; 2016. p. 177–85. Available from: http://dx.doi.org/10.1007/978-3-319-46547-0_19
86. Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z. DBpedia: A Nucleus for a Web of Open Data. In: *The Semantic Web* [Internet]. Springer Berlin Heidelberg; 2007. p. 722–35. Available from: http://dx.doi.org/10.1007/978-3-540-76298-0_52
87. Vrandečić D. Wikidata: a new platform for collaborative data collection. In: *Proceedings of the 21st International Conference on World Wide Web* [Internet]. New York, NY, USA: Association for Computing Machinery; 2012 [cited 2021 Nov 22]. p. 1063–4. (WWW '12 Companion). Available from: <https://doi.org/10.1145/2187980.2188242>

88. Baader F, Calvanese D, McGuinness D, Patel-Schneider P, Nardi D. The Description Logic Handbook: Theory, Implementation and Applications [Internet]. Cambridge University Press; 2003. 555 p. Available from: https://play.google.com/store/books/details?id=e6_hJtM07qwC
89. OWL 2 web ontology language profiles (second edition) [Internet]. [cited 2021 Aug 19]. Available from: <https://www.w3.org/TR/owl2-profiles/>
90. Hoehndorf R, Dumontier M, Oellrich A, Wimalaratne S, Rebholz-Schuhmann D, Schofield P, et al. A common layer of interoperability for biomedical ontologies based on OWL EL. *Bioinformatics* [Internet]. 2011 Apr 1;27(7):1001–8. Available from: <http://dx.doi.org/10.1093/bioinformatics/btr058>
91. OWL Collaboration. owltools [Internet]. Github; 2020 [cited 2021 Jul 16]. Available from: <https://github.com/owlcollab/owltools>
92. Alshahrani M, Khan MA, Maddouri O, Kinjo AR, Queralt-Rosinach N, Hoehndorf R. Neuro-symbolic representation learning on biological knowledge graphs. *Bioinformatics* [Internet]. 2017 Sep 1;33(17):2723–30. Available from: <http://dx.doi.org/10.1093/bioinformatics/btx275>
93. Babar MA, Zhu L, Jeffery R. A framework for classifying and comparing software architecture evaluation methods [Internet]. 2004 Australian Software Engineering Conference. Proceedings. 2004. Available from: <http://dx.doi.org/10.1109/aswec.2004.1290484>
94. Callahan TJ, Baumgartner WA, Hunter LE. Biomedical KG Construction Survey [Internet]. Zenodo; 2021. Available from: <https://zenodo.org/record/5790040>
95. Bastian M, Heymann S, Jacomy M. Gephi: An Open Source Software for Exploring and Manipulating Networks. *ICWSM* [Internet]. 2009 Mar 19 [cited 2021 Jul 28];3(1):361–2. Available from: <https://ojs.aaai.org/index.php/ICWSM/article/view/13937>
96. Martin S, Michael Brown W, Klavans R, Boyack KW. OpenOrd: an open-source toolbox for large graph layout. In: *Visualization and Data Analysis 2011* [Internet]. International Society for Optics and Photonics; 2011 [cited 2021 Jul 28]. p. 786806. Available from: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.871402>
97. Jackson RC, Balhoff JP, Douglass E, Harris NL, Mungall CJ, Overton JA. ROBOT: A Tool for Automating Ontology Workflows. *BMC Bioinformatics* [Internet]. 2019 Jul 29;20(1):407. Available from: <http://dx.doi.org/10.1186/s12859-019-3002-3>
98. Tripodi IJ, Callahan TJ, Westfall JT, Meitzer NS, Dowell RD, Hunter LE. Applying knowledge-driven mechanistic inference to toxicogenomics [Internet]. *bioRxiv*. 2019 [cited 2019 Sep 26]. p. 782011. Available from: <https://www.biorxiv.org/content/10.1101/782011v1>
99. Joslyn CA, Aksoy S, Callahan TJ, Hunter LE, Jefferson B, Praggastis B, et al. Hypernetwork Science: From Multidimensional Networks to Computational Topology [Internet]. *arXiv [cs.DM]*. 2020. Available from: <http://arxiv.org/abs/2003.11782>
100. Malec SA, Taneja SB, Witonsky KF, Shaaban CE, Karim HT, Levine AS, et al. Modeling Alzheimer's Disease by Combining Knowledge Extracted from Biomedical Literature with

Biomedical Ontologies [Internet]. Informatics Summit of the American Medical Informatics Association ; March 22-25 2021; Virtual. Available from: https://sanyabt.github.io/files/talks/PS01_AMIA_Malec.pdf

101. Taneja SB, Callahan T, Brochhausen M, Paine M, Kane-Gill S, Boyce R. Designing potential extensions from G-SRS to ChEBI to identify natural product-drug interactions [Internet]. 2021. Available from: <https://zenodo.org/record/5736386>
102. Cappelletti L, Fontana T, Casiraghi E, Ravanmehr V, Callahan TJ, Joachimiak MP, et al. GraPE: fast and scalable Graph Processing and Embedding [Internet]. arXiv [cs.LG]. 2021. Available from: <http://arxiv.org/abs/2110.06196>
103. Valentini G, Casiraghi E, Cappelletti L, Ravanmehr V, Fontana T, Reese J, et al. Het-node2vec: second order random walk sampling for heterogeneous multigraphs embedding [Internet]. arXiv [cs.LG]. 2021. Available from: <http://arxiv.org/abs/2101.01425>
104. HuBMAP Consortium. The human body at cellular resolution: the NIH Human Biomolecular Atlas Program. *Nature* [Internet]. 2019 Oct;574(7777):187–92. Available from: <http://dx.doi.org/10.1038/s41586-019-1629-x>
105. Reitz KM, Hall DE, Shinall MC Jr, Shireman PK, Silverstein JC. Using the Unified Medical Language System to expand the Operative Stress Score - first use case. *J Surg Res* [Internet]. 2021 Aug 28;268:552–61. Available from: <https://www.sciencedirect.com/science/article/pii/S0022480421004972>
106. UMLS-Graph: UMLS Graph database for semantic queries [Internet]. Github; [cited 2021 Dec 10]. Available from: <https://github.com/dbmi-pitt/UMLS-Graph>
107. ontology-api: The HuBMAP Ontology Service [Internet]. Github; [cited 2021 Dec 10]. Available from: <https://github.com/hubmapconsortium/ontology-api>
108. Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res* [Internet]. 2004 Jan 1;32(Database issue):D267–70. Available from: <http://dx.doi.org/10.1093/nar/gkh061>