




Data and text mining

ARAX: a graph-based modular reasoning tool for translational biomedicine

Amy K. Glen ^{1,†}, Chunyu Ma^{2,†}, Luis Mendoza³, Finn Womack², E. C. Wood ¹, Meghamala Sinha¹, Liliana Acevedo¹, Lindsey G. Kvarfordt¹, Ross C. Peene¹, Shaopeng Liu², Andrew S. Hoffman⁴, Jared C. Roach³, Eric W. Deutsch^{3,†}, Stephen A. Ramsey^{1,5,†} and David Koslicki ^{2,6,7,*†}

¹School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA, ²Huck Institutes of the Life Sciences, Pennsylvania State University, State College, PA 16802, USA, ³Institute for Systems Biology, Seattle, WA 98109, USA, ⁴Interdisciplinary Hub for Digitalization and Society, Radboud University, Nijmegen 6500GL, The Netherlands, ⁵Department of Biomedical Sciences, Oregon State University, Corvallis, OR 97331, USA, ⁶Department of Biology, Pennsylvania State University, State College, PA 16801, USA and ⁷Department of Computer Science and Engineering, Pennsylvania State University, State College, PA 16802, USA

*To whom correspondence should be addressed.

[†]The authors wish it to be known that, in their opinion, the first two authors and last three authors should be regarded as Joint Authors.

Associate Editor: Jonathan Wren

Received on August 15, 2022; revised on December 17, 2022; editorial decision on January 26, 2023; accepted on February 7, 2023

Abstract

Motivation: With the rapidly growing volume of knowledge and data in biomedical databases, improved methods for knowledge-graph-based computational reasoning are needed in order to answer translational questions. Previous efforts to solve such challenging computational reasoning problems have contributed tools and approaches, but progress has been hindered by the lack of an expressive analysis workflow language for translational reasoning and by the lack of a reasoning engine—supporting that language—that federates semantically integrated knowledge-bases.

Results: We introduce ARAX, a new reasoning system for translational biomedicine that provides a web browser user interface and an application programming interface (API). ARAX enables users to encode translational biomedical questions and to integrate knowledge across sources to answer the user's query and facilitate exploration of results. For ARAX, we developed new approaches to query planning, knowledge-gathering, reasoning and result ranking and dynamically integrate knowledge providers for answering biomedical questions. To illustrate ARAX's application and utility in specific disease contexts, we present several use-case examples.

Availability and implementation: The source code and technical documentation for building the ARAX server-side software and its built-in knowledge database are freely available online (<https://github.com/RTXteam/RTX>). We provide a hosted ARAX service with a web browser interface at arax.rtx.ai and a web API endpoint at [arax/v1.3/ui/](http://arax.rtx.ai/api/arax/v1.3/ui/).

Contact: dmk333@psu.edu

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

Structured biomedical knowledge is rapidly accumulating in primary databases such as ChEMBL (Mendez *et al.*, 2019), DrugBank (Wishart *et al.*, 2006), Reactome (Fabregat *et al.*, 2018), OBO Foundry (Smith *et al.*, 2007), KEGG (Kanehisa and Goto, 2000), Semantic MEDLINE Database (SemMedDB) (Kilicoglu *et al.*,

2012), UniProtKB (UniProt Consortium, 2021) and the Unified Medical Language System (UMLS) (Bodenreider, 2004). Using knowledge-based computational reasoning to answer translational questions such as *what drugs might be repurposed to treat Adams–Oliver syndrome?* requires integrating facts spanning a variety of concept types including drugs, targets, pathways, genetic variants, phenotypes and diseases. To facilitate such computational

reasoning, there have been numerous efforts to integrate knowledge from various biomedical databases using knowledge graph (KG) abstractions (Joubert et al., 1998a, b; Sowa, 1992) consisting of a labeled multigraph in which each node represents a concept and each edge represents a concept–concept relationship, that is, a ‘triple’. Several such biomedical KGs have been described (Dumontier et al., 2014; Himmelstein et al., 2017; Messina et al., 2018a; Morton et al., 2019; Piñero et al., 2017; Sanders et al., 2020; github.com/gnn4dr/DRKG), including RTX-KG2, which we developed and described previously (Wood et al., 2021) and which integrates all of the aforementioned primary databases with a semantic layer that is described by the open-standard Biolink model (Unni et al., 2022). Given such a comprehensive KG with a unified semantic layer, a biomedical question can be transformed into a *query*

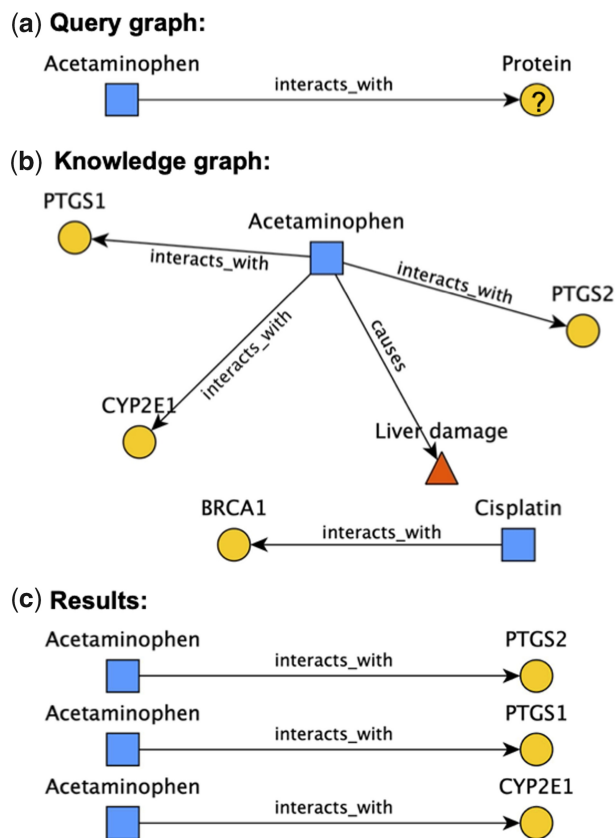


Fig. 1. An example query graph (a) and the set of subgraphs that ‘fulfill’ it (c) from an example knowledge graph (b). Item a depicts a query graph that asks for proteins that interact with the drug acetaminophen. Query graphs are connected graphs that define particular *patterns* to search for in-graph data. In this example, the acetaminophen query node is considered ‘pinned’, because it is constrained to a specific concept; Biolink-defined (Unni et al., 2022) node categories (`Protein`) and edge predicates (`interacts_with`) constrain the rest of the query. While this query graph is quite simple, there is no hard limit to how many nodes/edges query graphs can include (though there may be a practical limit, due to the tendency for query runtime and memory consumption to increase with query graph size), and they may contain cycles and/or be multigraphs. Item (b) depicts a hypothetical KG that might be used to answer this query. This KG contains three different categories of nodes (represented by node shape/color) and two different edge predicates. Item (c) shows the three subgraphs of the KG that fulfill the query graph. A subgraph fulfills the query graph if it fits the pattern of the query graph in terms of its structure and constraints, with acetaminophen connected to a protein via an `interacts_with` edge. This example is simplified for clarity’s sake, but in reality there are more complexities to what it means to fulfill a query graph. For instance, Biolink node categories and edge predicates are hierarchical in nature, meaning a valid result for the example query graph could, for instance, use the edge predicate `physically_interacts_with`, because that predicate is a descendant of `interacts_with`. In addition, one could specify *multiple* node categories or edge predicates on a single node/edge in their query graph; in that case, nodes/edges with *either* of those categories/predicates are considered to fulfill that query node/edge

graph (Yih et al., 2015) [which represents a search pattern, analogous to a `select` statement in the SPARQL language (Angles and Gutierrez, 2008); see Fig. 1 for more details] and/or a *graph analysis workflow* in order to generate a list of ‘answers’/‘results’, each corresponding to a *subgraph* of the KG. To facilitate that process for translational applications, an open-source, web-based and standards-based tool is needed for designing, expressing, executing and refining KG analysis workflows.

Efforts to develop a comprehensive platform for expressing, executing and refining KG analysis workflows have used new query languages, web application programming interface frameworks, computational reasoning paradigms, graph topology heuristics and machine-learning methods. BIOZON (Birkland and Yona, 2006) provides a SQL-based query and search function for its KG; Hetionet (Himmelstein et al., 2017) integrates multiple databases into a single graph for drug repurposing and leverages graph path-finding; BioGraph (Messina et al., 2018b) uses the Gremlin graph query language; SPOKE (Nelson et al., 2019) provides capabilities for finding a concept of interest and visualizing its ‘node neighborhood’; BioThings Explorer (Xin et al., 2017) builds a biomedical KG by querying SmartAPI-registered application programming interfaces (APIs); ROBOKOP (Morton et al., 2019) constructs a query-specific KG which it analyzes to identify candidate answers and mediKanren (Byrd et al., 2020) uses the miniKanren logic programming system for drug repurposing applications. Despite much progress, two key problems remain: (1) how to provide control and transparency of the graph analysis workflow that is executed to answer a user’s question and (2) how to rank potential answers. The control/transparency problem entails a trade-off: on one extreme, a general-purpose graph query language such as Neo4j Cypher (neo4j.com/developer/cypher) provides complete control, but expressing, modifying and extending a state-of-the-art drug repositioning analysis workflow is extremely complex and demanding for many users. On the other extreme, canned analysis workflow approaches in practice have limited customizability. In addition, many tools have a limited number of knowledge sources and/or lack a standardized semantic layer with sufficient expressiveness.

1.1 ARAX overview and key advantages

Here, we describe ARAX (arax.rtx.ai), a new computational reasoning tool for querying and exploring biomedical knowledge and data. ARAX provides three key advantages versus previous systems:

1. ARAXi, ARAX’s intuitive language for specifying a workflow for analyzing a KG (Section 2.1);
2. access to around 40 knowledge providers (KPs) (which themselves access over 100 underlying knowledge sources) from a single reasoning tool, using a standardized interface and semantic layer (Section 2.2) and
3. a versatile method for scoring search result subgraphs (Section 2.1.5).

Together, these advantages are designed to facilitate and improve reasoning-based answering of translational questions (Austin, 2018) such as *what genetic conditions protect against asthma?* *What drugs target proteins associated with the cyclooxygenase pathway?* and *How can expression of KCNMA1 be pharmacologically inhibited?*

ARAX is one of six reasoning engines in the Biomedical Data Translator system (henceforth, ‘Translator’) (Translator Consortium, 2019a, b), (see the ‘Funding’ section), a distributed computational system for accelerating translational science. Translator has a layered architecture; queries are interpreted by a coordinating service, which sends the query to the reasoning engines which, in turn, consult registered KP services to answer the query. All translator services exchange messages using a standard web interface, the translator reasoner API (TRAPI) (github.com/NCATSTranslator/ReasonerAPI), whose technical specification is collaboratively maintained. For its semantic layer, TRAPI uses the biolink model (github.com/biolink/biolink-model) to specify types of biological entities and relationships. ARAX can be directly queried via either its TRAPI API or its web browser

user interface; these provide various ways to formulate queries and explore results (Section 2.3).

A typical query and corresponding workflow ARAX may use to answer it is depicted in Figure 2, along with a diagram of ARAX’s architecture. Its architecture is modular, with five different analysis modules that can be run using ARAXi commands (Section 2.1). ARAX is mainly built on top of RTX-KG2 (Wood *et al.*, 2021), our large-scale biomedical KG that has a TRAPI-compliant query interface and is registered as a translator KP; but ARAX also queries close to 40 other KPs with specific areas of emphasis (Section 2.2.1), including genome-wide association study data, electronic health record data from medical centers and molecular data from high-throughput cellular assays.

As illustrated in Figure 2, ARAX accepts input queries in any of three representations: a TRAPI query graph, ARAXi or TRAPI operations. TRAPI query graphs are graph-based templates representing the user’s question (see Fig. 1), expressed in JavaScript Object Notation (JSON) format. The ARAXi language, which can be used to express a custom graph analysis workflow including overlaying of annotations and/or filtering of edges to reduce the result-set size, is described in Section 2.1. TRAPI operations (github.com/NCATSTranslator/OperationsAndWorkflows), the third way of specifying a graph analysis workflow, is an ARAXi-inspired, translator-standardized vocabulary; it is translated into ARAXi by the ARAX interpreter.

2 Materials and methods

2.1 ARAXi KG analysis language

To address the need for an intuitive language for expressing KG analysis workflows, we developed ARAXi, a procedural language that allows a user or software tool to concisely express biomedical questions. The five main ARAXi modules (see Sections 2.1.1–2.1.5) can be used individually or combined in a graph analysis workflow (Fig. 2). Each ARAXi command corresponds to an analysis module; this modular design simplifies both construction and reuse of KG analysis workflows.

All input queries—whether in the form of a TRAPI query graph, ARAXi or TRAPI operations—are ultimately translated into a series of ARAXi commands, which together define the knowledge-retrieval and analysis workflow for answering the query. In particular, two core ARAXi commands, `add_qnode` and `add_qedge`, are used to incrementally construct a query graph, which can then be expanded by the `ARAX_expander` module (Section 2.1.1) into a query-specific KG; the query-specific KG can then be refined or have knowledge overlaid using the four other ARAX modules (Sections 2.1.2–2.1.5).

2.1.1 Expand a KG: ARAX_expander

The `ARAX_expander` module uses all TRAPI-compliant KPs registered in the SmartAPI registry (Zaveri *et al.*, 2017) (see Section 2.2 for more details) to find subgraphs that satisfy the input query graph. For each edge in the query graph, `ARAX_expander` determines which KPs are capable of fulfilling that query edge (based on metadata provided dynamically by the KPs’ APIs) and then queries those KPs concurrently via their TRAPI-compliant APIs. When querying each KP, `ARAX_expander` converts any node identifiers in the query to semantically equivalent identifiers from controlled vocabularies that the KP prefers. This is accomplished by using the ARAX Node Synonymizer service (Section 2.2.2). ARAX combines answers returned from the KPs into a single answer KG that is ‘canonicalized’, meaning that it does not contain semantically redundant nodes.

`ARAX_expander` processes the query graph in a breadth-first fashion; that is, it retrieves all answers from KPs for a given query edge before moving on to the next query edge. It avoids combinatorial explosion by pruning down the answer set that it receives for each query edge in an adaptive and predictive manner: it uses the `ARAX_overlay` module’s Fisher Exact Test function (Section 2.1.2) combined with `ARAX_ranker` (Section 2.1.5) to decide which triples to retain for the given query edge. By default, the number of triples

to retain for each query edge (the ‘pruning threshold’) is dynamically determined according to the projected degree of combinatorial explosion the given query edge will produce, based on heuristics pertaining to the depth of categories and predicates in the Biolink hierarchy. Currently, these heuristics essentially set the threshold to one of three values when one of the query edge’s nodes is ‘unpinned’ (not constrained to specific concepts): 100 if the given query edge is unconstrained in terms of categories and predicates or if the root Biolink categories/predicates are used, 200 if a category known to be very prevalent in KPs is used (this generally only includes categories one generation away from the root node in the Biolink category tree; e.g. `biolink: ChemicalEntity`) and 500 otherwise. Doubly pinned queries are given a higher prune threshold of 5000, due to their inherent constraint to specific concepts. Alternatively, a user may specify a per-query prune threshold that overrides the dynamically determined default pruning via the `ARAXi prune_threshold` parameter. Having to prune answers after expanding each query edge is the main downside to the breadth-first approach of processing the query graph, since good answers may be lost; while a depth-first approach would avoid this, it allows for less batch querying of KPs (i.e. sending queries in which one node is pinned to many concepts) and thus results in prohibitively slow query times in our experience. In the future, we plan to improve the dynamic pruning method to do query backtracking in some form; that is, if no answers for the next query edge are found based on the set of answers that were retained for the prior query edge, try using more of the prior query edge’s answers to see if they produce answers for the next query edge and so on. This will effectively result in a sort of hybrid breadth-first/depth-first approach that may combine the best of both worlds.

Because users have different preferences in terms of the amount of time they are willing to wait for ARAX to provide answers to their query, `ARAX_expander` also provides a KP timeout parameter, which specifies how long `ARAX_expander` should wait for a response from each KP. Users who are more interested in getting answers quickly can specify a short (e.g. 30s) timeout, while a user whose use-case prioritizes comprehensiveness over expediency can use a longer KP timeout.

2.1.2 Overlay/annotate contextual information: ARAX_overlay

The `ARAX_overlay` module enhances the `ARAX_expander`-built KG by overlaying ‘virtual edges’ (which are virtual in the sense that they do not fulfill edges present in the original query graph) to denote (1) structural similarity (i.e. a statistically significant number of shared neighbors) of two nodes in the KG; (2) a predicted ‘treats’ relation between drug and disease nodes via a link-prediction model or (3) statistically significant co-association in a database such as a clinical database, a combined clinical–epidemiological database or the biomedical literature. Currently, the overlay function (which is called with the `overlay` command in ARAXi) can overlay nodes or edges with the following seven kinds of contextual information:

Fisher exact test evaluates how significant/non-random a one-hop connection is between a given list of subject nodes and an object node based on the P -value calculated by following the traditional Fisher’s exact statistical test. This is equivalent to a phenotype or gene enrichment analysis in statistical genetics, but for arbitrary node categories. Given a set $\{v_s\}$ of subject nodes and an object node v_o , a P -value is calculated to indicate how significant/non-random an edge is between the subjects and object. This requires using the following 2×2 contingency table as input to the ‘stats-fisher_exact’ method in the Scipy python package (Virtanen *et al.*, 2020):

	Number of nodes in $\{v_s\}$	Number of nodes not in $\{v_s\}$
Number of nodes connected to v_o	a	b
Number of nodes not connected to v_o	c	d

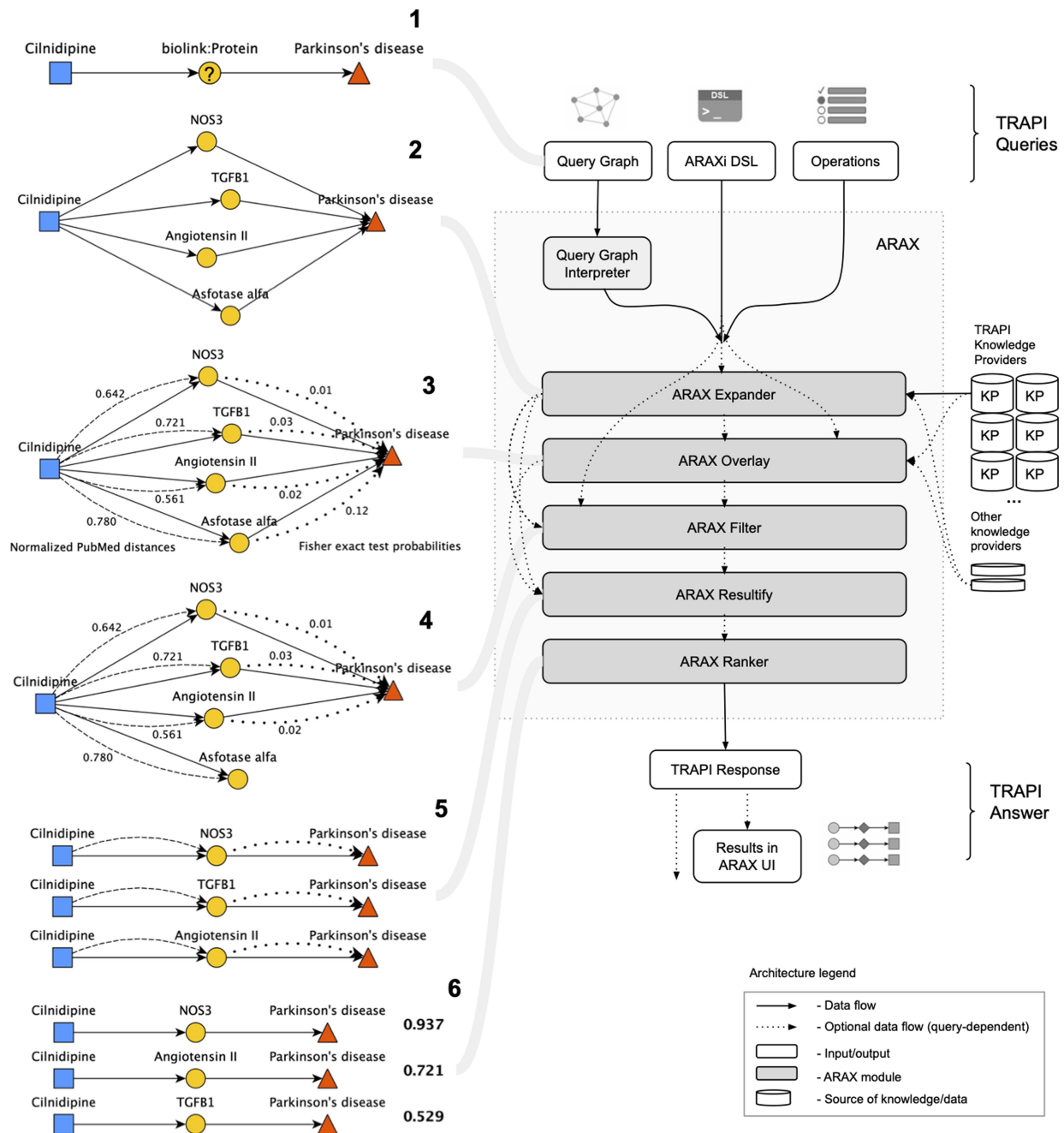


Fig. 2. ARAX answers queries using *workflows*, in which each step is handled by a particular ARAX module. Left panel: The left half of this figure depicts a typical simple query that might be submitted to ARAX (Step 1) and how ARAX goes about answering that query (Steps 2–6). More specifically, Step 1 shows a visual representation of the query in query graph form, which asks for proteins associated with both Parkinson’s disease and the drug cilnidipine. Step 2 depicts the Expand step, which reaches out to various KPs for answers to the given query and combines their answers into an answer knowledge graph. Step 3 represents the Overlay step, in which the answer KG is overlaid/annotated with contextual quantitative information in ‘virtual’ edges (dashed/dotted lines). Step 4 depicts the Filter step, in which nodes and edges are filtered from the answer KG based on the statistical information overlaid in Step 3. Step 5 represents the Resultify step, which finds all of the subgraphs in the answer KG that fulfill the query graph. Step 6 represents the Rank step, which calculates a score for each result subgraph indicating the overall degree of epistemic support for that answer. Right panel: The right half of this figure depicts ARAX’s architecture, which is centered around five modules, each of which corresponds to a typical workflow step. Notably, ARAX accepts queries in three different forms, all of which are based on the TRAPI data model: (1) a query graph (as shown in Step 1), (2) the ARAXi graph analysis workflow definition language (Section 2.1) or (3) TRAPI operations. The ARAX system translates the two non-ARAXi input forms into an ARAXi workflow that specifies the series of ARAX modules that will be run to answer the query. The five core ARAX modules (Expander, Overlay, Filter, Resultify and Ranker) operate independently and may be combined in various orders depending on the query. When the input is in the form of a query graph, the Query Graph Interpreter layer selects an appropriate series of ARAXi commands to answer the question at hand; this selection of ARAXi commands is done using a template-based system in which templates were manually curated by examining common query graph structures or archetypes. The final answer knowledge graph and results are returned in a TRAPI JSON response either directly to the requesting software program or to the ARAX UI for browser rendering (Section 2.3.1)

where ‘connected’/‘not connected’ means with respect to the KG RTX-KG2 when considering nodes of the same category as v_o . $\{a, b, c, d\}$ are the integer values representing number of nodes.

Jaccard similarity measures how many shared intermediate nodes with a specific category (e.g. ‘Protein’) can be found via knowledge sources between a subject node and an object node.

For example, this can be used to find drugs that interact with many genes associated to a disease; find genes associated with many phenotypes of a disease or other such ‘many intermediate connections’ queries. More rigorously, given a starting node v_s connected to n intermediate nodes $\{v_i\}_{i=1}^n$ which each connects to an ending node v_e , the Jaccard similarity between v_s and v_e is defined as follows, with $\text{OutDeg}(v)$ representing the out-degree of a node in the KG:

$$\text{JS}(v_s, v_e) = \frac{n}{\text{OutDeg}(v_s)}. \quad (1)$$

Drug-treats-disease probability predicts the probability that a given drug can treat a given disease. We developed this computational drug repurposing approach based on the model framework proposed by [Womack et al. \(2019\)](#). In order to adapt this model to our core KG RTX-KG2, we modified the implementation of the model by replacing `node2vec` ([Grover and Leskovec, 2016](#)) with `GraphSage` ([Hamilton et al., 2017](#)) for generating the node embeddings as well as using direct concatenation instead of the Hadamard product to generate the input features for a Random Forest classifier for drug–disease pairs. Additional performance gains may be realized through the use of alternative embedding or prediction models, though Section 3.3 indicates the current approach performs favorably.

Normalized PubMed distance measures the significance of co-occurrence of pairs of terms (corresponding to node names) in article abstracts in the PubMed (i.e. MEDLINE) database, using the ‘normalized Google distance’ (NGD) measure ([Cilibrasi and Vitanyi, 2004](#))

$$\text{NGD}(t_1, t_2) = \frac{\max\{\log f(t_1), \log f(t_2)\} - \log f(t_1, t_2)}{\log N - \min\{\log(t_1), \log(t_2)\}}, \quad (2)$$

where t_1 and t_2 are the biomedical terms (node names) used in RTX-KG2; $f(t_1)$ and $f(t_2)$, respectively, represent the total number of unique PubMed IDs associated with this term; $f(t_1, t_2)$ is the number of unique and shared PubMed IDs between t_1 and t_2 and N is the number of pairs of article abstract and MeSH term ([Rogers, 1963](#)) annotations in PubMed.

PubMed abstracts obtained by mapping node identifiers to MeSH terms and obtaining the PubMed articles associated with this MeSH term.

Columbia Open Health Data (COHD) Clinical Information ([Ta et al., 2018](#)) obtained from electronic health records of 1.7 M patients during a 5-year period. For a given pair of terms, three statistical measures are obtained from COHD: observed clinical frequencies; the log-ratio between observed and expected counts and the χ^2 test statistic.

Exposure data statistical association data for pairs of terms in health records and between clinical terms and environmental exposure variables in epidemiological databases are obtained through the Integrated Clinical and Environmental Exposures Services (ICEES) service ([Fecho et al., 2019](#)), which obtains data from University of North Carolina Health; the Drug Induced Liver Injury Network and the National Institute of Environmental Health Sciences Personalized Environment and Genes Study.

2.1.3 Filter the KG: ARAX_filter

Multi-hop queries can produce such large answer KGs that it is sometimes useful to prune down the KG to remove less important nodes or edges before proceeding. The `ARAX_filter` module allows one to do this; it provides several options for selectively removing nodes/edges based on their contextual information (often quantitative information added by `ARAX_overlay`) according to user-defined thresholds. This functionality can be invoked using the `filter_kg` ARAXi command. `ARAX_filter` also provides various methods for limiting the number of results returned and for sorting the results created by `ARAX_resultify` (see Section 2.1.4), for example, by score, edge/node counts or particular edge/node attributes.

2.1.4 Create results: ARAX_resultify

The `ARAX_resultify` module finds and returns all subgraphs in the answer KG that fulfill the query graph; this step is typically performed after the answer KG has been constructed and (optionally) pruned. ARAXi exposes a boolean `ignore_edge_direction` parameter that controls whether `ARAX_resultify` should ignore or enforce edge direction when determining whether a subgraph fulfills the query graph. When enforcing edge direction, `ARAX_resultify` will only return subgraphs for which edge directions match those of the query graph edges they are fulfilling. When ignoring edge direction, KG edges are allowed to fulfill query graph edges in the reverse direction (meaning, `PTGS2-interacts_with->Acetaminophen` would be a valid result for the query graph `Acetaminophen-interacts_with->Protein`).

2.1.5 Score and rank result graphs: ARAX_ranker

The `ARAX_ranker` module assigns a score between 0 and 1 (where a higher score represents a higher confidence answer) to each result graph and orders the results by descending score. Scores are computed as follows:

1. For each ‘result graph’:
 - a. Each edge in the result graph is assigned a score $[0,1]$, as follows:
 - i. Any scalar attribute (e.g. Fisher exact test P -value or Jaccard index) on the edge is normalized to a value between 0 and 1 using a sigmoid function parameterized for the specific type of attribute.
 - ii. For any edges originating from SemMedDB, the attribute containing PubMed references for publications supporting the given assertion is converted into a scalar value via the function:

$$\left[1 + \left(\frac{b}{n}\right)^a\right]^{-1}, \quad (3)$$
 where n represents the number of publications listed in the attribute. The values of $a = \log_2(9)$ and $b = 4$ were chosen so that four supporting publications result in a value of 0.5 (i.e. 50% confidence) and eight publications give a value of 0.9.
 - iii. The given edge’s normalized, weighted scalar values are then multiplied to create one single score between 0 and 1 that is assigned to that edge.
 - b. Using the normalized, weighted edge scores from Step 1a, three graph weights (calculated using max-flow, Frobenius norm and weight-of-longest-geodesic-path) are obtained for the result graph.
2. Results are then ranked in descending order by their three metric scores calculated in Step 1b (in three separate lists).
3. Each result is assigned a final score that is the average of its three ranks from Step 2 (normalized between 0 and 1).

The ARAX system automatically runs `ARAX_ranker` after `ARAX_resultify`, but `ARAX_ranker` can also be run individually via the `rank_results` ARAXi command; in principle, a list of result subgraphs produced by another translator tool could be ranked using this ARAX service.

2.2 Knowledge/data

2.2.1 Richness

ARAX currently uses around 40 different KPs to answer queries; the exact count depends on the number of TRAPI-compliant KPs registered in SmartAPI (smart-api.info/registry) at query runtime, since `ARAX_expander` dynamically selects from all such KPs for a given

Table 1. ARAX’s top 25 KPs in terms of number of meta-triples they can answer queries about

KP	Cat.	Pred.	Meta.
RTX-KG2	57	86	45 300
CAM-KP	105	50	28 601
Ontology-KP	81	14	4434
Automat-robokop	20	190	2296
Automat-uberongraph	18	66	1414
Service Provider (by BioThings Explorer)	22	97	1102
Automat-ctd	16	91	730
SRI Reference KG	14	89	644
Text Mined Cooccurrence API	18	1	322
Automat-cord19	17	6	291
MolePro (Molecular Provider)	20	126	288
Automat-biolink	17	13	210
Automat-hetio	15	27	159
Automat-ontology-hierarchy	18	2	134
Automat-drug-central	11	17	104
Automat-pharos	13	12	95
Automat-hmdb	10	9	72
Automat-human-go	10	23	60
Automat-icees-kg	8	1	57
COHD (Columbia Open Health Data)	5	2	50
SPOKE KP	18	24	44
Automat-gtopdb	6	11	40
Automat-viral-proteome	10	11	35
Automat-panther	8	7	24
Automat-gtex	2	20	20

Notes: Cat., categories; Pred., predicates; Meta., meta-triples. More information on each KP along with links to their APIs are available in the SmartAPI registry (footnote 5).

query. The KPs themselves obtain information from various knowledge sources, giving ARAX access to a combined total of more than 100 knowledge sources (e.g. UniProt, DrugBank, DisGeNET, ChEMBL, MONDO and SemMedDB). Calculating the exact count of underlying knowledge sources is difficult due to the lack of such information in structured form for some KPs, but between RTX-KG2’s 73 knowledge sources (Wood et al., 2021), SPOKE’s approximately 40 knowledge sources (spoke.ucsf.edu/data-tools) and the BioThings Explorer Service Provider’s 32 integrated sources (github.com/biothings/biothings_explorer), ARAX has access to at least 100 distinct underlying knowledge sources (we note that there is some overlap in knowledge sources between KPs). The true count of underlying knowledge sources is likely notably higher since ARAX uses many other KPs that were not included in the above estimate.

In total, ARAX’s KPs can answer queries of about 135 different categories of nodes (e.g. biolink: Protein, biolink: Disease) and 281 different edge predicates (e.g. biolink: interacts_with, biolink: treats). This results in a total of 76 443 distinct meta-triples (combinations of subject node category, edge predicate and object node category) that ARAX has access to. The breakdown of these counts by KP is shown in Table 1 for ARAX’s top 25 KPs in terms of number of supported meta-triples. Figure 3 shows a small selection of ARAX’s overall meta-graph, including some of the most commonly queried node categories. The counts for each KP are based on their TRAPI v1.3.0 API/meta_knowledge_graph endpoints, accessed on December 12, 2022. All such APIs are listed in the SmartAPI Registry online 5.

2.2.2 Knowledge federation

Two aspects of ARAX’s architecture are crucial for knowledge federation: its adherence to standards for query APIs and for its

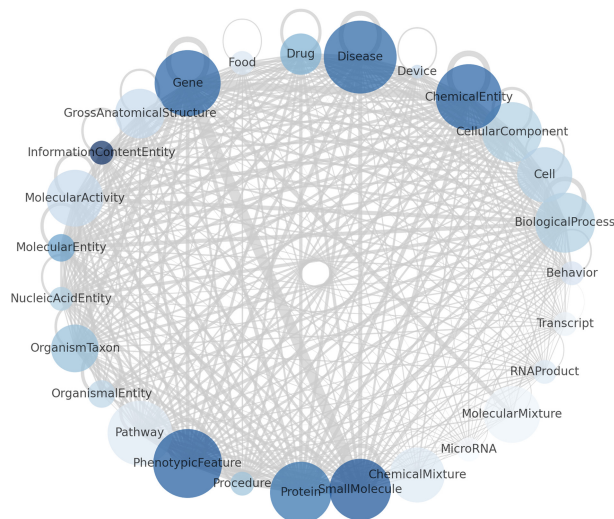


Fig. 3. A small portion of ARAX’s meta-graph including a selection of the most commonly queried node categories. The size of each node represents the number of ARAX’s KPs that can answer queries involving that category, the color of each node represents the number of node identifier types ARAX has access to (through its KPs) for that category and the thickness of each edge represents the number of distinct predicates ARAX has access to between two nodes with the given categories

semantic layer (i.e. TRAPI and Biolink, respectively) and its comprehensive service for mapping between equivalent node identifiers (i.e. node synonymization).

2.2.3 TRAPI and the biolink model. ARAX bypasses many of the data integration challenges that arise when combining information from multiple sources by adhering to a standard web API format (TRAPI; github.com/NCATSTranslator/ReasonerAPI) that itself adheres to a standard semantic format (Biolink: github.com/biolink/biolink-model). Not only does ARAX’s API conform to TRAPI, but in the course of answering queries, ARAX largely only uses KPs that themselves speak TRAPI [The only exceptions are two in-house KPs that are stored locally and queried using SQL: Drug-Treats-Disease and Normalized PubMed Distance (Section 2.1.2).] This means that ARAX has to do very minimal processing to relay queries to and combine answers from different KPs, since they all represent their answers in the same format, using the same node categories and edge predicates. Because of this, any KP that speaks TRAPI can be plugged into the ARAX_expander module with ease. In fact, ARAX_expander (Section 2.1.1) dynamically selects TRAPI KPs that are registered in the SmartAPI registry (Zaveri et al., 2017), meaning ARAX is able to add new KPs without any human intervention.

2.2.4 ARAX node synonymizer. Because there are many overlapping controlled vocabularies within biomedicine, concepts (or the nodes that represent them) can often be described using multiple identifiers. For instance, MONDO:0019391, DOID:13636 and ORPHANET:84 are all valid identifiers for the disease Fanconi anemia, coming from the Monarch Disease Ontology (Mungall et al., 2017), the Disease Ontology (Schriml et al., 2019) and the Orphanet Rare Disease Ontology (Vasant et al., 2014; Weinreich et al., 2008), respectively. Different KPs may refer to the same concept using different identifiers, making it challenging to integrate their results. To address this problem, we created the ARAX node synonymizer, a node synonym mapping service built into ARAX. The node synonymizer determines node equivalencies by combining four kinds of evidence:

1. Concept equivalence information provided by a translator web service called the Standards and Reference Implementation Node Normalizer (github.com/TranslatorSRI/NodeNormalization),

- Nodes connected by `biolink:same_as` relations in RTX-KG2,
- Nodes with identical names and
- Node semantic type compatibility.

The node synonymizer uses this information to partition a given set of node identifiers into sets of semantically equivalent identifiers. Each of these clusters is assigned a single representative concept identifier from among the cluster's members. ARAX uses the node synonymizer to map all node identifiers returned from KPs to these canonical identifiers, facilitating merging of the KPs' responses.

2.3 Interfaces for accessing ARAX

2.3.1 ARAX web browser interface

The ARAX web browser user interface ('ARAX UI'; `arax.rtx.ai`) provides an intuitive, human-friendly mechanism for querying ARAX and exploring the answers it returns. It allows users to formulate biomedical questions in four different formats: an interactive visual query graph builder, TRAPI operations, TRAPI JSON and ARAXi. The interactive query graph builder allows users to construct a query graph in a visual fashion via drop-down lists and clickable buttons, with no JSON or TRAPI knowledge required. The TRAPI operations input aids users in creating a TRAPI-compliant workflow by providing a drop-down list of allowable operations and guided entry of operation parameters. The TRAPI JSON input method facilitates sharing and re-use of graph analysis workflows. The ARAXi input method (described in Section 2.1) provides users with easy-to-understand syntax to formulate more complex query workflows, along with drop-down menus of ARAXi commands to facilitate discovery. For each query posted to the ARAX UI, the output includes five sections:

- Summary summarizes the answers to the query in a simple table sorted by ARAX-defined scores that can be considered a measure of confidence (Section 2.1.5).
- KG provides a visual representation of the answer KG for easy viewing of its topology.
- Results provide a display of all the results that satisfy the input query graph, ranked as in Section 2.1.5. Clicking on individual results displays an interactive graphical representation of the result. Within each result, clicking on nodes and edges displays detailed information including evidence/provenance information and concept descriptions.
- Messages show all messages logged during processing of the query, including any errors or warnings.
- Provenance shows the number of edges in the final KG by predicate type and knowledge source(s) that they came from. (May require refreshing the web browser page to display data.)

The ARAX UI provides persistent URLs that enable fast retrieval of results from previously run queries. Apart from these main functions, the ARAX UI also provides three additional tools/services:

- Synonym lookup exposes the ARAX Node Synonymizer (described in 'ARAX Node Synonymizer' in Section 2.2.2); a user can input a concept identifier or name and see its set of equivalent identifiers and other synonym information.
- Dev Info provides a JSON-formatted view of UI-server communications for development purposes.
- System activity displays the status and activity of previous queries submitted to ARAX over a user-selectable time interval and retrieves corresponding queries and results (if they completed successfully).

2.3.2 ARAX API

ARAX can also be used via its publicly accessible API (`arax.rtx.ai/api/arax/v1.3/openapi.json`), which is listed in the SmartAPI registry (Zaveri *et al.*, 2017). The ARAX API includes `/query` and `/asyncquery` endpoints that accept TRAPI input, as well as an `/entity` endpoint that exposes the ARAX Node Synonymizer service, allowing users to programmatically retrieve equivalent identifiers and other synonym information for a given concept.

3 Use cases

To illustrate how ARAX can be used to explore biomedical knowledge, we present three different use cases: two focused queries, one to search for molecular mechanisms in bipolar disorder and one to search for the mechanism of action for an antiviral for COVID-19 disease, and a larger-scale analysis concerning drug repurposing.

3.1 Use Case 1: Bipolar disorder

Bipolar disorder is a mental health condition that affects over 46 M people worldwide (Ferrari *et al.*, 2016). It is characterized by mood swings that can include excessive happiness (mania) and/or excessive sadness (depression). The causes of bipolar disorder are not well understood, but are thought to include both genetic and environmental factors, with stress and substance abuse contributing to disease severity. It has been recently proposed (Hasin *et al.*, 2021) that decreased expression of D-amino acid oxidase (DAO) in cerebellar neurons might increase the risk for bipolar disorder by affecting the regulation of N-methyl-D-aspartate receptors (NMDARs). Using the ARAX web browser interface, we explored the connection between DAO and bipolar disorder via protein intermediaries in the context of a two-hop query graph (e.g. 'DAO—*—bipolar disorder' where * represents a peptide/protein mediator). Within the browser interface, we posted the ARAXi commands of a two-hop query graph to ARAX, which returned 17 result subgraphs (Fig. 4). The query and its results are available as the [Supplementary Material](#) (`github:RTXteam/RTX/notes/vignettes.md#bipolar-disorder`). The 'NMDARs' result [which we expected to see based on prior literature (Hasin *et al.*, 2021)] is in the top 10 highest-scoring results, based on ARAX's result-graph ranking algorithm (see Section 2.1.5). For some of the other peptide/protein results returned, ARAX provided hyperlinks to publications that support the relevant relations in the specific DAO-to-bipolar result graph such as the amino acid proline (Fig. 5).

3.2 Use Case 2: Coronavirus disease 2019

Coronavirus disease 2019 (COVID-19), caused by the SARS-CoV-2 pathogen, has resulted in a worldwide pandemic. Researchers have made extensive efforts (Grundeis *et al.*, 2023; Hassanipour *et al.*, 2021; Reis *et al.*, 2022; Rosas *et al.*, 2021; Temple *et al.*, 2021) to repurpose existing drugs to treat COVID-19. Remdesivir was the

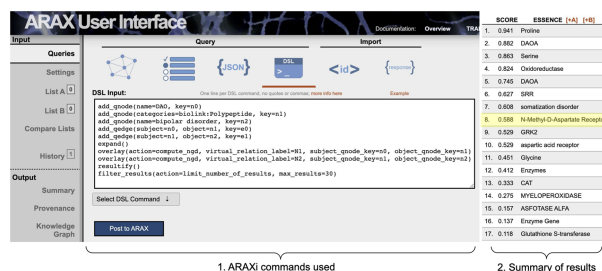


Fig. 4. ARAX Web UI showing the ARAXi workflow (1) used for Use Case 1 (exploring what proteins might be involved in the connection between DAO and bipolar disorder) and the 17 returned results with their corresponding ranking scores (2), as they appear in the Summary tab in the UI. The expected result of 'NMDARs' is highlighted in yellow

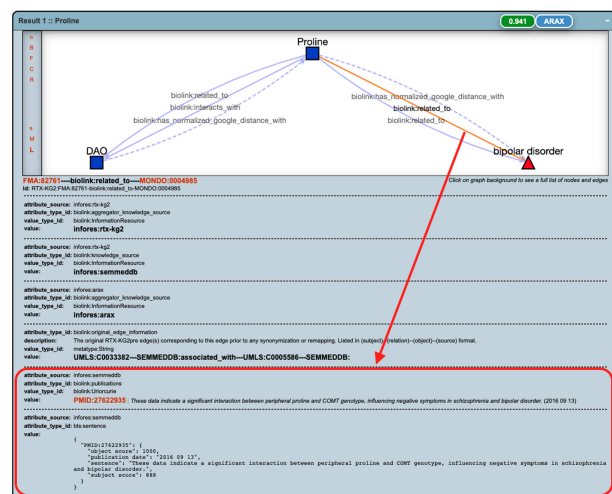


Fig. 5. A ‘result graph’ for a single result (‘proline’) for the two-hop query for mediators between DAO and bipolar disorder, with supporting publication (PMID: 27622935) and its corresponding relevant sentence shown (highlighted by a red square) for a specific edge ‘Proline—biolink: related_to—bipolar disorder’

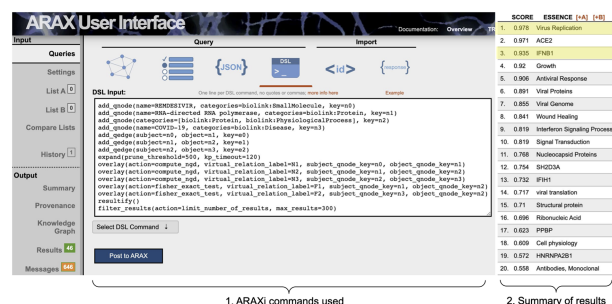


Fig. 6. ARAXi workflow (1) for finding the remdesivir mechanism-of-action downstream of RdRp for treating COVID-19 (Use Case 2), along with the top 25 returned results (2). The results of ‘Virus Replication’ and ‘IFNB1’ mentioned in the main text are highlighted in yellow

first repurposed antiviral drug that was approved by the United States Food and Drug Administration for the treatment of COVID-19. Remdesivir was originally developed to treat hepatitis C and then found to be effective against many other viral diseases including Ebola and other coronavirus diseases. Its mechanism of action is to interfere with the viral RNA-dependent RNA polymerase (RdRp) (Eastman *et al.*, 2020). By using ARAX, we can provide a series of potential three-hop biological-relation paths between remdesivir and COVID-19 via RNA-directed RNA polymerase. Figure 6 shows the top 25 ARAX results with the corresponding ARAXi commands. The query and its results are available as the [Supplementary Material](#) (github:RTXteam/RTX/notes/vignettes.md#covid-19). A top-ranking result, ‘Virus Replication’, is precisely the mechanism of action that has been reported by various researchers (Eastman *et al.*, 2020; Kocic *et al.*, 2021; Malin *et al.*, 2020). Furthermore, ARAX results can reveal potential downstream mediators of therapeutic efficacy such as release of type I interferon, an antiviral cytokine which SARS-CoV-2 infection is known to inhibit (Wang *et al.*, 2021) (see result ‘IFNB1’ for this query).

3.3 Use Case 3: Recovering drug/disease relationships

Drug repurposing is a strategy for finding treatments for diseases by searching for and validating new indications for existing (i.e. already approved) drugs. Computational approaches to this task often involve developing a link-prediction model: a model that, when given a drug and disease, predicts the probability that the drug treats the

disease. One such model, given in Section 2.1.2, is accessible via ARAX. Here, we describe the performance of that model in predicting ‘treats’ edges in the RTX-KG2 KP. Importantly, this model was not trained on RTX-KG2 ‘treats’ edges but rather data from SemMedDB (Kilicoglu *et al.*, 2012), MyChem (Xin *et al.*, 2018) and NDF-RT (Brown *et al.*, 2004) (as described in Womack *et al.*, 2019), though some RTX-KG2 ‘treats’ edges may exist in these three training datasets. (Namely, those edges from SemMedDB, which is one of RTX-KG2’s sources; MyChem and RTX-KG2 also have some overlapping sources, such as DrugCentral.) In RTX-KG2, there are 266k `biolink:treats` edges. We used the model in Section 2.1.2 to predict, for all such 266k drug/disease (Note here that ‘drug’ means `biolink:Drug` or `biolink:SmallMolecule`, and ‘disease’ means `biolink:Disease` or `biolink:PhenotypicFeature`) pairs, the probability that a ‘treats’ relationship exists between them. We then selected all (drug, (‘non-treats’ relation), disease) triples, where ‘non-treats’ relation is any relation besides the Biolink predicates: `treats`, `prevents`, `treated_by`, `ameliorates`, `is_ameliorated_by`, `related_to` or `disrupts`. We used the model to calculate the link prediction probabilities of all 184k such ‘non-treats’ edges. Figure 7 shows the distributions of this link prediction task, demonstrating the model can successfully distinguish between drug/disease pairs with the ‘treats’ or ‘not-treats’ relationship. Using a cutoff probability of 0.8, 40% of existing RTX-KG2 ‘treats’ relationships were recovered, while only 17% of ‘non-treats’ edges were above this cutoff.

4 Discussion

The purpose of ARAX and Translator is to enable integrated analysis of structured biomedical knowledge in order to provide ranked, coherent answers to translational biomedical questions. ARAX’s three key innovations—ARAXi, its approximately 40 KPs, and its result subgraph ranking algorithm—together provide significant leverage to enable a researcher to tackle translational questions that are more complex and rely more on basic science (and not just on clinically validated) knowledge. Further, ARAX’s web browser interface enables composing queries and browsing/exploring results, without having to programmatically post-process results or query the API.

While it shares the goal of leveraging knowledge to advance translation, ARAX is not an AI-based Biomedical Question-Answering system such as IBM Watson (Ferrucci, 2012), MedQA (Yu *et al.*, 2007) or BioSQUASH (Shi *et al.*, 2007). Such question-answering systems can provide a direct semantic answer computationally extracted from relevant biomedical documents via natural language processing (NLP) techniques. Thus, the knowledge used in these systems is only limited to document-based knowledge. Further, ARAX differs from the NLP-based document search tool BioMed Explorer (sites.research.google/biomedexplorer/); BioMed Explorer is designed to answer a question by finding relevant resources (such as articles) using semantic search and showing excerpts from those resources. ARAX, in contrast, is based on multiple types of data sources (e.g. publications, databases and electronic health records). Instead of directly providing an intuitive answer, ARAX is more like a ‘searching and computing engine’ where it provides a self-developed domain language to translate a specific biomedical question into a query graph, ‘search’ for answers from different KPs and then perform computations with the resulting graphs. Thus, it can facilitate querying and exploring a significant fraction of published biomedical knowledge and public biomedical knowledge-bases. However, unlike search engines or semantic search engines, ARAX allows users to create custom KG analysis workflows, as well as access reasoning modules for specific translational applications like drug repurposing.

ARAX was built to help biomedical researchers explore structured knowledge and help generate new hypotheses. ARAX is for research purposes and is not meant to be used by clinicians in the course of treating patients. As the system is being actively developed, there is no expectation that results from queries run on ARAX will be retained indefinitely. Future enhancements to ARAX will include

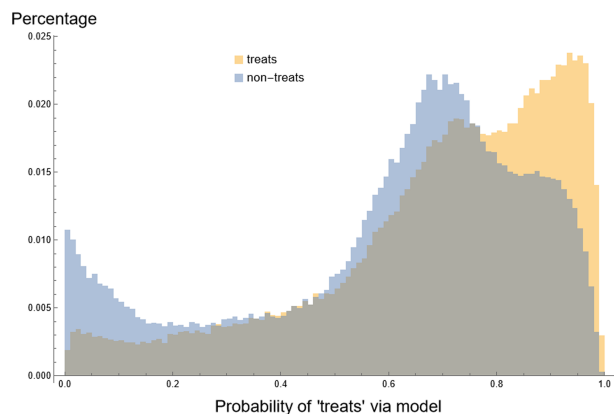


Fig. 7. Overlaid histograms of the link prediction probability for the 'treats' relationship for all drug/disease pairs having a biolink: treats edge between them (in yellow) or non-treats edge between them (in blue)

increasing the Node Synonymizer's accuracy and improving drug-treats-disease prediction via alternative embedding models and classifiers.

5 Conclusion

ARAX is a computational reasoning tool that allows users to easily extract, explore and analyze knowledge from diverse biomedical resources. With self-developed ARAXi, ARAX allows users to encode their complex biomedical questions into specific KG analysis workflows in an intuitive manner. These workflows can intelligently integrate data from multiple biomedical KPs, analyze and filter that data, and rank the final results. The ARAX UI further simplifies this process by providing multiple query options and facilitating interactive exploration of query results. The ARAX API enables incorporating ARAX's knowledge retrieval and reasoning capabilities within a workflow involving non-ARAX tools. We believe that ARAX can help life sciences researchers to more effectively interpret new research findings and develop new hypotheses.

Acknowledgements

We thank Mark Williams, Tyler Beck, Noel Southall, Christine Colvis, Sarah Stemann, Debbi Adelakun, Melissa Haendel, Chris Bizon, Karamarie Fecho, Patrick Wang, Chris Mungall, Sierra Moxon, Matt Brush, Paul Shannon, Andrew Su, Chunlei Wu, Kevin Xin, Will Byrd, Andrew Crouse, John Osborne, Jeff Henrikson, Greg Rosenblatt, Sergio Baranzini, Matt Might, Arnab Nandi, Sui Huang and Liang Huang for advice and/or feedback on ARAX. We also thank Dac-Trung Nguyen, Deqing Qu, Tim Yoon, Tim Putman, Richard Bruskiwicz, Pouyan Ahmadi, Ujjval Kumaria, Jason McClelland, Yao Yao, Zheng Liu, David Palzer, Sarah Oliphant, Boris Cao, Ke Wang, Max Wang, Jamie Slome, Kanna Bhargav Chevva and Daniel Lin for technical assistance. The authors thank Amazon Web Services for in-kind computing infrastructure support.

Funding

Support for this work was provided by NCATS through NIH awards OT2TR003428 and OT2TR002520. Any opinions expressed in this document are those of the Translator community at large and do not necessarily reflect the views of NIH, NCATS, individual Translator team members or affiliated organizations and institutions. A.K.G. gratefully acknowledges support from the ARCS Foundation.

Conflict of Interest: none declared.

References

Angles,R. and Gutierrez,C. (2008) The Expressive Power of SPARQL. In: *The Semantic Web - ISWC 2008*. Springer, pp. 114–129.

- Austin,C.P. (2018) Translating translation. *Nat. Rev. Drug Discov.*, 17, 455–456.
- Birkland,A. and Yona,G. (2006) BIOZON: A system for unification, management and analysis of heterogeneous biological data. *BMC Bioinformatics*, 7, 70.
- Bodenreider,O. (2004) The unified medical language system (UMLS): Integrating biomedical terminology. *Nucleic Acids Res.*, 32, D267–70.
- Brown,S.H. *et al.* (2004) VA national drug file reference terminology: A cross-institutional content coverage study. *Stud. Health Technol. Inform.*, 107, 477–481.
- Byrd,W. *et al.* (2020) mediKanren: A system for bio-medical reasoning. In: *miniKanren Workshop*.
- Cilibrasi,R. and Vitanyi,P.M.B. (2004) The Google similarity distance. *arXiv preprint arxiv:cs/0412098*.
- Dumontier,M. *et al.* (2014) Bio2RDF release 3: A larger connected network of linked data for the life sciences. In: *Proceedings of the 2014 International Conference on Posters & Demonstrations Track*, Vol. 1272. Citeseer, pp. 401–404.
- Eastman,R. *et al.* (2020) Remdesivir: A review of its discovery and development leading to emergency use authorization for treatment of COVID-19. *ACS Cent. Sci.*, 6, 672–683.
- Fabregat,A. *et al.* (2018) Reactome graph database: Efficient access to complex pathway data. *PLoS Comput. Biol.*, 14, e1005968.
- Fecho,K. *et al.* (2019) A novel approach for exposing and sharing clinical data: The translator integrated clinical and environmental exposures service. *J. Am. Med. Inform. Assoc.*, 26, 1064–1073.
- Ferrari,A. *et al.* (2016) The prevalence and burden of bipolar disorder: Findings from the global burden of disease study 2013. *Bipolar Disord.*, 18, 440–450.
- Ferrucci,D.A. (2012) Introduction to “this is Watson”. *IBM J. Res. Dev.*, 56, 1:1–1:15.
- Grover,A. and Leskovec,J. (2016) node2vec: Scalable feature learning for networks. *KDD*, pp. 855–864.
- Grundeis,F. *et al.* (2023) Remdesivir for the treatment of COVID-19. *Cochrane Database Syst. Rev.*, 1.
- Hamilton,W.L. *et al.* (2017) Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.*, 30.
- Hasin,N. *et al.* (2021) A rare variant in D-amino acid oxidase implicates NMDA receptor signaling and cerebellar gene networks in risk for bipolar disorder. *medRxiv preprint medrxiv:2021.06.02.21258261*.
- Hassanipour,S. *et al.* (2021) The efficacy and safety of favipiravir in treatment of COVID-19: A systematic review and meta-analysis of clinical trials. *Sci. Rep.*, 11(1).
- Himmelstein,D. *et al.* (2017) Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6, e26726.
- Joubert,M. *et al.* (1998a) Review of biomedical knowledge and data representation with conceptual graphs. *Methods Inf. Med.*, 37, 86–96.
- Joubert,M. *et al.* (1998b) UMLS-based conceptual queries to biomedical information databases: An overview of the project ARIANE. Unified Medical Language System. *J. Am. Med. Inform. Assoc.*, 5, 52–61.
- Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 28, 27–30.
- Kilicoglu,H. *et al.* (2012) Semmeddb: A pubmed-scale repository of biomedical semantic predications. *Bioinformatics*, 28, 3158–3160.
- Kokic,G. *et al.* (2021) Mechanism of SARS-CoV-2 polymerase stalling by remdesivir. *Nat. Commun.*, 12(1), 279.
- Maiella,S. *et al.* (2013). Orphanet and its consortium: where to find expert-validated information on rare diseases. *Revue Neurologique*, 169, S3–8.
- Malin,J. *et al.* (2020) Remdesivir against COVID-19 and other viral diseases. *Clin. Microbiol. Rev.*, 34(1), e00162–20.
- Mendez,D. *et al.* (2019) ChEMBL: Towards direct deposition of bioassay data. *Nucleic Acids Res.*, 47, D930–D940.
- Messina,A. *et al.* (2018a) BioGrakn: A knowledge graph-based semantic database for biomedical sciences. In: Barolli, L. and Terzo, O. (eds) *Advances in Intelligent Systems and Computing*. Springer International Publishing, Cham, pp. 299–309.
- Messina,A. *et al.* (2018b) BioGraph: A web application and a graph database for querying and analyzing bioinformatics resources. *BMC Syst. Biol.*, 12(5), 75–89.
- Morton,K. *et al.* (2019) ROBOKOP: An abstraction layer and user interface for knowledge graphs to support question answering. *Bioinformatics*, 35, 5382–5384.
- Mungall,C. *et al.* (2017) The Monarch Initiative: An integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Res.*, 45, D712–D722.

- Nelson, C. et al. (2019) Integrating biomedical research and electronic health records to create knowledge-based biologically meaningful machine-readable embeddings. *Nat. Commun.*, 10(1), 3045.
- Piñero, J. et al. (2017) DisGeNET: A comprehensive platform integrating information on human disease-associated genes and variants. *Nucleic Acids Res.*, 45, D833–D839.
- Reis, G. et al.; TOGETHER Investigators. (2022) Effect of early treatment with fluvoxamine on risk of emergency care and hospitalisation among patients with COVID-19: The TOGETHER randomised, platform clinical trial. *Lancet Glob. Health*, 10, e42–e51.
- Rogers, F.B. (1963) Medical subject headings. *Bull. Med. Libr. Assoc.*, 51, 114–116.
- Rosas, I. et al. (2021) Tocilizumab in hospitalized patients with severe COVID-19 pneumonia. *N. Engl. J. Med.*, 384, 1503–1516.
- Sanders, G. et al. (2020) Topological analysis of the SPOKE graph. *Technical report*, U.S. Department of Energy. doi:10.2172/1669224
- Schriml, L. et al. (2019) Human disease ontology 2018 update: Classification, content and workflow expansion. *Nucleic Acids Res.*, 47, D955–D962.
- Shi, Z. et al. (2007) Question answering summarization of multiple biomedical documents. *Advances in Artificial Intelligence*. Springer, pp. 284–295.
- Smith, B. et al. (2007) The OBO foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.*, 25, 1251–1255.
- Sowa, J.F. (1992) Conceptual graphs as a universal knowledge representation. *Comput. Math. Appl.*, 23, 75–93.
- Ta, C. et al. (2018) Columbia open health data, clinical concept prevalence and co-occurrence from electronic health records. *Sci. Data*, 5(1), 1–17.
- Temple, C. et al. (2021) Toxic effects from ivermectin use associated with prevention and treatment of COVID-19. *N. Engl. J. Med.*, 385, 2197–2198.
- Translator Consortium. (2019a) Toward a universal biomedical data translator. *Clin. Transl. Sci.*, 12, 86–90.
- Translator Consortium. (2019b) The biomedical data translator program: Conception, culture, and community. *Clin. Transl. Sci.*, 12, 91–94.
- UniProt Consortium. (2021) UniProt: The universal protein knowledgebase in 2021. *Nucleic Acids Res.*, 49, D480–D489.
- Unni, D. et al. (2022) Biolink model: A universal schema for knowledge graphs in clinical, biomedical, and translational science. *Clin. Transl. Sci.*, 15(8): 1848–1855.
- Virtanen, P. et al.; SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nat. Methods*, 17, 261–272.
- Wang, W. et al. (2021) SARS-CoV-2 nsp12 attenuates type I interferon production by inhibiting IRF3 nuclear translocation. *Cell. Mol. Immunol.*, 18, 945–953.
- Weinreich, S. et al. (2008) Orphanet: A European database for rare diseases. *Ned. Tijdschr. Geneesk.*, 152, 518–519.
- Wishart, D. et al. (2006) DrugBank: A comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res.*, 34, D668–D672.
- Womack, F. et al. (2019) Leveraging distributed biomedical knowledge sources to discover novel uses for known drugs. *bioRxiv preprint biorxiv:765305*.
- Wood, E. et al. (2021) RTX-KG2: A system for building a semantically standardized knowledge graph for translational biomedicine. *bioRxiv preprint biorxiv:464747*.
- Xin, J. et al. (2017) Biothings Explorer: utilizing JSON-LD for linking biological APIs to facilitate knowledge discovery. *F1000Research*, 6.
- Xin, J. et al. (2018) Cross-linking biothings APIs through JSON-LD to facilitate knowledge exploration. *BMC Bioinformatics*, 19, 30.
- Yih, W.-T. et al. (2015) Semantic parsing via staged query graph generation: Question answering with knowledge base. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, pp. 1321–1331.
- Yu, H. et al. (2007) Development, implementation, and a cognitive evaluation of a definitional question answering system for physicians. *J. Biomed. Inform.*, 40, 236–251.
- Zaveri, A. et al. (2017) smartAPI: Towards a more intelligent network of Web APIs. In: Blomqvist, E. et al. (eds) *The Semantic Web*. Springer International Publishing, Cham, pp. 154–169.