EXPLORATION OF DOMAIN-SPECIFIC KNOWLEDGE GRAPHS FOR TESTABLE HYPOTHESIS GENERATION

Daniel R Korn

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2022

Approved by:

Stanley Ahalt

Rada Chirkova

Colin Raffel

Natalie Stanley

Alexander Tropsha

ABSTRACT

DANIEL R KORN: Exploration of Domain-Specific Knowledge Graphs For Testable Hypothesis Generation
(Under the direction of Alexander Tropsha and Rada Chirkova)

In the span of a decade, we have brought about a fundamental shift in the way we structure, organize, store, and conceptualize biomedical datasets. Data which had previously been siloed has been gathered, organized, and aggregated into central repositories, interlinked with each other by categorizing these vast sums of knowledge into well defined ontologies. These interlinked databases, better known as knowledge graphs, have come to redefine our ability to explore the current state of our knowledge, answer complex questions about how objects relate to each other, and invent novel connections in vastly different research disciplines.

With these knowledge graphs, new ideas can be quickly formulated, instead of relying upon the insight of a single scientist or small team of experts, these ideas can be made leveraging the vast historical catalog of research progress that has been captured in biomedical databases. Knowledge graphs can be used to propose hypotheses which narrow the nearly infinite array of possible explorations which can link any pair of ideas to only those which have some historical and practical considerations. In this way, we hope to utilize these knowledge graphs to produce hypotheses, promote those which are viable, and provide them to biomedical experts.

In this work, we aim to develop methodologies to produce meaningful hypotheses using these graphs as inputs. We approach this problem by (i) utilizing intrinsic mathematical properties of the intermediate nodes along pathways, (ii) translating existing biomedical ideas into graphical structures, and (iii) incorporating niche domain-specific biomedical datasets to explore domain problems. We have shown the ability of these methods to produce practical and useful hypotheses and pathways which can be utilized by experts for immediate exploration.

"The most merciful thing in the world, I think, is the inability of the human mind to correlate all its contents. We live on a placid island of ignorance in the midst of black seas of infinity, and it was not meant that we should voyage far. The sciences, each straining in its own direction, have hitherto harmed us little; but some day the piecing together of dissociated knowledge will open up such terrifying vistas of reality, and of our frightful position therein, that we shall either go mad from the revelation or flee from the light into the peace and safety of a new dark age."

– H.P. Lovecraft

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADRD	Alzheimer's Disease and Related Dementia
AOP	Adverse Outcome Pathway
API	Application Programming Interface
COP	Clinical Outcome Pathway
CTD	Comparative Toxicogenomics Database
DREAM	Drug Repurposing for Effective Alzheimer's Medicines
FDA	Federal Drug Administration
GWAS	Genome-Wide Association Studies
HGNC	HUGO Gene Nomenclature Committee
HPO	Human Phenotype Ontology
MRR	Mean Reciprocal Rank
KG	Knowledge Graph
SMILES	Simplified Molecular-Input Line-Entry System
ROBOKOP	Reasoning Over Biomedical Objects linked in Knowledge Oriented Pathways

CHAPTER 1

Introduction

Accessing, storing, and exploring large amounts of interconnected data has problems that have been actively studied in computer science. The creation and querying of massive relational databases are a field of intense interest and research, frequently allowing users to find relational information from a database nearly instantly. However, these relational databases are not as effective when users have extremely diverse data, with highly complex relationships between different data types.(Robinson et al., 2015)

Biologists, chemists, geneticists, and many other branches of life and medical scientists have worked for many centuries building up the scientific knowledge surrounding the human body. Each new generation slowly builds up on the discoveries and knowledge of the previous, collecting new data, discovering new compounds, building more elaborate and precise tools to complete experiments. It is through this slow deliberate exploration that we as a species have conquered plagues, endemic diseases which were once seen as a death sentence like tuberculosis can now be treated with a variety of readily available drugs.

Two biomedical papers are submitted to the PubMed repository every minute, with more than a million papers being published each year, and even keeping up with the novel research in a specialized field can be a full-time task for researchers (Landhuis, 2016). As the amount of research increases, it becomes less likely for any one researcher to be able to understand the entirety of new research, even just in their specialized field, let alone in the broader domains. One major issue with biomedical research is the siloization of results and the repeating of existing studies due to the lack of discoverability of results (Denton et al., 2021; Rodriguez-Esteban, 2022).

1.1 Knowledge Graphs

Accessing, storing, and exploring large amounts of interconnected data has are problems that have been actively studied in computer science. The creation and querying of massive relational databases is a field of intense interest and study; frequently allowing users to nearly instantly find relational information from a database. For many generations, these databases have served as the solution to nearly every large-scale data processing problem in the field. However, these relational databases are not as effective when users have extremely flexible data, with highly diverse relationships between different data types (Webber, 2012).

When processing data which may need to express very dynamic linkages between complex data types which are often updated, a graph database may be more desirable. Well maintained graph database software, such as Neo4J(Team, 2022), SPARQL(Harris et al., 2013), and Apache TinkerPop(Rodriguez, 2015), have already seen a large amount of use in both industrial and academic settings. These databases differ by seperating data into two broad categories, objects (nodes) and relationships between those objects (edges). These nodes may have an arbitrary number of labels and properties attached to them, enable large flexibility in the types of data that can be represented. Logically representing data as nodes within that graph, and edges as the linkages between data (Hogan et al., 2021; Robinson et al., 2015).

An ontology is a conceptual level description of a particular domain, encompassing the level of detail that information should take and of ways in which different entities can relate with each other(Ehrlinger and Wöß, 2016). In knowledge graphs, the choice of ontology and of how the knowledge is represented is critical. Nodes in the knowledge graph represent ideas and entities from the real world; a knowledge graph may contain a node for the University of North Carolina at Chapel Hill (UNC-Chapel Hill), an instance of a university that should serve as the node's class. Similarly, edges in knowledge graphs serve to represent how these entities interact, i.e., how they capture facts and statements about relationships between entities. For instance, we can represent the

statement ”UNC-Chapel Hill is located in North Carolina” as two nodes: UNC-Chapel Hill and North Carolina, joined via an edge which states their relationship, in this case, located in.

The idea of one large dataset collecting all information in the world is a powerful one. We can see formulations of this idea from the initial foundations of the internet, with the notion of a semantic web powered by XML and RDF being prevalent in the 1990s(Berners-Lee et al., 2001). This semantic web aimed to provide one universal way to link all the data being produced all over the internet. By interlinking different datasets together, this web would create a unified dataset that encapsulates all knowledge. Ultimately, the semantic web failed to succeed, the semantic web has received criticism for multiple reasons. 1) Its use being too niche, 2) the logistical difficulties of getting so many large and disparate organizations to agree on a common ontology, 3) lack of computational power and storage to handle such a large database.(Hogan, 2020) Still, the idea of modern knowledge graphs and graph database architecture has its roots in the philosophy of the semantic web project.

It was not until truly massive datasets with diverse nodes and complex relationships became practical to collect and construct that the utility of graph databases became more apparent. We can view the rise of knowledge graphs as parallel with big data, which is defined as the ability to rapidly collect, capture, store, and distribute large volumes of information.(Gandomi and Haider, 2015) The age of ”big data” began sometime in the early 2010’s. However, the exact date is contentious, as corporations, government agencies, and non-profits all began pivoting goals to information aggregation and dissemination.(Gandomi and Haider, 2015) Without this abundance of data, the creation of large knowledge graphs would still be as infeasible as it was when attempted during the age of the semantic web. Other changes also needed to be done to enable knowledge graphs. Organizations were created whose sole purpose was to create and maintain knowledge graphs, allowing maintainers to act as the foremost authority. These maintainers have the final say in: (1) The target domain of a specific knowledge graph, and (2) The level of ontological rigor the database would have. Additionally, the National Institutes of Health (NIH) created the National Center for Advancing Translational Sciences (NCATS), which oversees and standardizes nascent

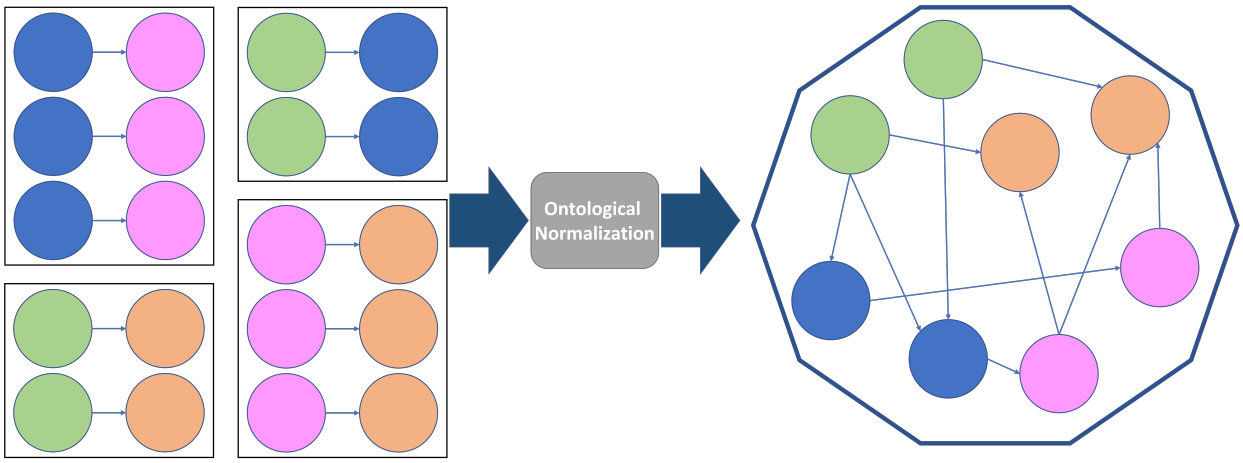


Figure 1.1: This figure shows a simplified version of the creation of a knowledge graph. The color of each node represents a different ontological category (for example, blue nodes could represent diseases, and pink nodes represent symptoms, etc). On the left most side of the picture, we present four databases each consisting of connections for how objects of a singular ontological type relate to another ontological class (an example of such a database may be Malacards, which provides information on how the presence of genes relates to occurrence of genetic diseases). In the middle we have an ontological normalization step, in which all nodes are brought into the same vocabulary. And finally the end result is a graph of interconnected nodes, uniting the information from each of the databases.

work on heterogeneous biomedical data integration.(Austin, 2016) These advances have created an environment where large-scale domain-specific knowledge graphs have gone from impossible to real, allowing the development of extremely powerful tools.

Many other private and non-profit organizations have developed valuable and exciting knowledge graphs in the last decade. Google leverages an internal private knowledge graph called ”Knowledge Vault” to provide semantic results for web searches.(Dong et al., 2014) The Knowledge Vault claims to have 45 million entities and 271 million ”confident facts” captured about these entities. YAGO (Yet Another Great Ontology) is a knowledge graph produced by Max-Planck-Institute in Germany.(Suchanek et al., 2007) The graph mines connections from WordNet and Wikipedia, producing extensive collections of facts and entities from community-sourced documentation. Wikidata is produced by the Wikimedia Foundation, a non-profit organization devoted to the creation and preservation of human knowledge.(Malyshev et al., 2018) Wikidata

Figure 1.2: Here we show the Google ”Knowledge Panel” for Fred Brooks, the founder of the UNC Computer Science Department. This panel was generated by Googling the query “Who is Fred Brooks?” and the aim of knowledge panels are to present a high level overview of a topic. This panel was generated from the internal “Knowledge Vault”(Dong et al., 2014) graph at Google.(Google, 2021b,a)

enables users to create and append facts about entities and knowledge from the world; it presently has 61 million entities and 750 million facts that relate to these entities.(Waagmeester et al., 2020)

Google’s “Knowledge Panels” are possible the most publicly available use of knowledge graphs. By taking advantage of their Knowledge Vault;(Dong et al., 2014) Google is able to provide fast answer to user questions without directing them to another website.(Google, 2021b,a) These panels show up on the in response to user queries which can be answered with factual information. In Figure 1.2 we show an example of one of Google’s Knowledge Panels for the Fred Brooks, who invented the 8-bit byte and founded the UNC Department of Computer Science in 1964, among many other life achievements. This panel was created in response to a user search of the question “Who is Fred Brooks?”

Although the terms ”graph database” and ”knowledge graph” are sometimes used interchangeably in the literature, there are many important distinctions between the two. Graph databases refer to the functional creation of data in a system with some graph structure. This structure may be utilized for fast retrieval and complex queries. On the other hand, knowledge graphs are collections of specific information put together in a graph structure. Knowledge graphs can often be powered by a graph database software, such as Neo4J(Webber, 2012), but may also be released as flat text files. For example, Amazon's Drug Repurposing Knowledge Graph (DRKG)(Ioannidis et al., 2020) was released as a series of RDF. Another trait of knowledge graphs is that they often provide confidence values for their connections, typically values between 0 (low confidence) and 1 (absolute confidence). In a traditional database, uncertain data are considered highly undesirable and must be purged and cleaned, leaving a dataset of 100% accurate information. Conversely, this uncertainty of linkage is a key trait of knowledge graphs, where information on what is true and how ideas relate is often open to interpretation. These confidence values enable users to make complex judgements on how different nodes may relate to each other.

Many other use cases of knowledge graphs exist. Siemens employs knowledge graphs to improve their engineering and manufacturing practices.(Hubauer et al., 2018) Cyber security has also benefitted greatly from the use of knowledge graphs, and graph databases in general. SEPSES (Semantic Processing of Security Event Streams) provides an interlinked repository of cybersecurity attacks, vulnerabilities, and known intrusions.(Kiesling et al., 2019) Graph databases have also played a key role in the evolving field of fraud detection, enabling both heuristic and machine learning based approaches to flag suspicious activity.(Jiang et al., 2019; Sadowski and Rathle, 2014) Knowledge graphs have also been created interlinking researchers, research interests, and historical discoveries.(Oldman and Tanase, 2018)

With the large amount of data for a specific domain area that has been accumulated, the challenge becomes to derive practical knowledge from these data. Certain graph databases serve a critical function as data retrieval systems, serving as a repository to enable users to collect and analyze known information for a specific question or quickly retrieve all the current information

and relationships for some queried entity. But to domain experts, it is often important to find novel connections within the data. The question can be stated bluntly "What can we derive from this database?" These users seek to go beyond simply utilize graph databases as knowledge repositories and instead create novel discoveries by leveraging existing data. We hope that reformulate the introduction of knowledge graphs will enable more complex handling of data .

Some initiatives have been created to help aid in the creation and organization of data. *FAIR Principles*, which stands for **F**indability, **A**ccessibility, **I**nteroperability, and **R**eusability, aim to be a general purpose (Wilkinson et al., 2016). As stated by Wilkinson et al. in the outline of these principles: "Good data management is not a goal in itself, but rather is the key conduit leading to knowledge discovery and innovation, and to subsequent data and knowledge integration and reuse by the community after the data publication process." Issues with how results of academic endeavors are shared with the world, typically through a publication, do not facilitate easy transfer of knowledge or ability to integrate results into other database. The FAIR project hopes to enforce database rules and guidelines on all government funded research projects. Similar efforts have been created in Europe, such as Open PHACTS (Pharmacological Concept Triple Store), a project which hopes to unite the discoveries of academic and industry into a centralized database (Williams et al., 2012). The aim is to create a data commons which information from the public sector, the private sector, and individuals are all entered into and accessible to everyone.

1.2 Biomedical Knowledge Graphs

General knowledge graphs have great potential as general question answer tools. As discussed above, Google utilizes knowledge graphs in their search algorithm to provide high quality answers to user queries, and they have been utilized for various other fields, such as cyber security and engineering.

The field of bioinformatics encompasses dozens of disciplines and further within those, hundreds of subdisciplines.(Can, 2014; Luscombe et al., 2001) Biology, chemistry, genetics, proteomics, statistics, and many other fields all play a critical role in the continued discovery and advancement

of human medical knowledge. Specialists from each of these groups may find discoveries specific to their skill set and purview which inspire and enable those in other disciplines. This feedback loop of different disciplines pushing knowledge in other areas forward is critical for the complex realm of the human body.

There exists a large number of biomedical databases in the public domain right now which provide detailed, accurate, and high-quality information for different information. These databases are often extremely specific to a subdiscipline of the biomedical field. An example of one such database is MetalPDB, this database contains information purely on how specific metals interact with proteins in the human body.(Andreini et al., 2013; Putignano et al., 2018) The narrow focus of these databases on a niche problem space could be viewed to create silos of information. But the breadth of knowledge is so massive, keeping a narrow focus enables maintainers of these databases to maintain high levels of quality and avoid errors which may occur from tracking information which they have no expertise in.

Other biomedical databases which are helpful to show to explore the field are ChEMBL (available at `https://www.ebi.ac.uk/chembl/`).(Gaulton et al., 2017; Mendez et al., 2012) The ChEMBL database is a collection of information on over two million chemical compounds. Each compound is given an ontological identifier unique to the ChEMBL database; for example "aspirin" has the identifier **CHEMBL25** (`https://www.ebi.ac.uk/chembl/compound_report_card/CHEMBL25/`). Within the ChEMBL database, each of these identifiers can be queried to provide information on existing knowledge surrounding the chemical. Some of the fields ChEMBL provides are:

1. **Synonyms** – A list of all names which may refer to the same chemical. Includes specific brand names (such as Viagra, Prozac, Wellbutrin) and generic names.

2. **FDA Approval Status** – The drug's approval by the Federal Drug Administration (FDA). This includes what phase of clinical trial a drug has been tried in.

3. **SMILES String** – The simplified molecular-input line-entry system (SMILES) string are character based representation of chemical structure.(Weininger, 1988)

4. **Drug Indications** – The medical conditions a drug has been documented to treat.

5. **Drug Mechanisms** – How a drug is believed to function, this topic is covered in greater deal in Section 3.2 (Clinical Outcome Pathways).

Some of these fields provide linkages to other databases. For each drug indication a compound has, ChEMBL provides not only information on what indication a drug treats, when this was discovered, and what sources provide it's information, it also provides a linkage to the Medical Subject Headings ontology (MeSH)(Lipscomb, 2000). To further our example, in the drug indications of **CHEMBL25** (aspirin), we can find an entry for Fever with **D005334** the identifier for fever in the MeSH Ontology (`https://id.nlm.nih.gov/mesh/D005334.html`).

Ontological identifiers for biomedical entities, such as ChEMBL and MeSH identifiers, provide the foundation for our biomedical knowledge graphs, serving as the nodes in the graph. The linkages between these datasets through interaction of these entities; such as in the above example **CHEMBL25** (aspirin) *treats* **D005334** (fever) enable us to connect nodes through biomedically relevant edges.

We present in Figure 1.3 an example of a biomedical knowledge graph. This example comes from the ROBOKOP biomedical knowledge graph (Section 1.5.1). In this example we see an interleaving of various different node types and relationships. In this graph we see an example of drug $\rightarrow$ gene connection, drug $\rightarrow$ disease connections, gene $\rightarrow$ biological process, biological process $\rightarrow$ disease. Each edge in the graph provides a label describing the connection between the two nodes, and in Neo4J extra properties may be added to edges.

Omics data aims to encapsulate all branches of science which end with the phrase "-omics", such as genomics, proteomics, epigenomics, *etc*. The aims of such a standard would be to facilitate simplistic sharing over broad categories of research and researchers (Chervitz et al., 2011). Historically these standards have been scattered and spotty, localized to academic groups without

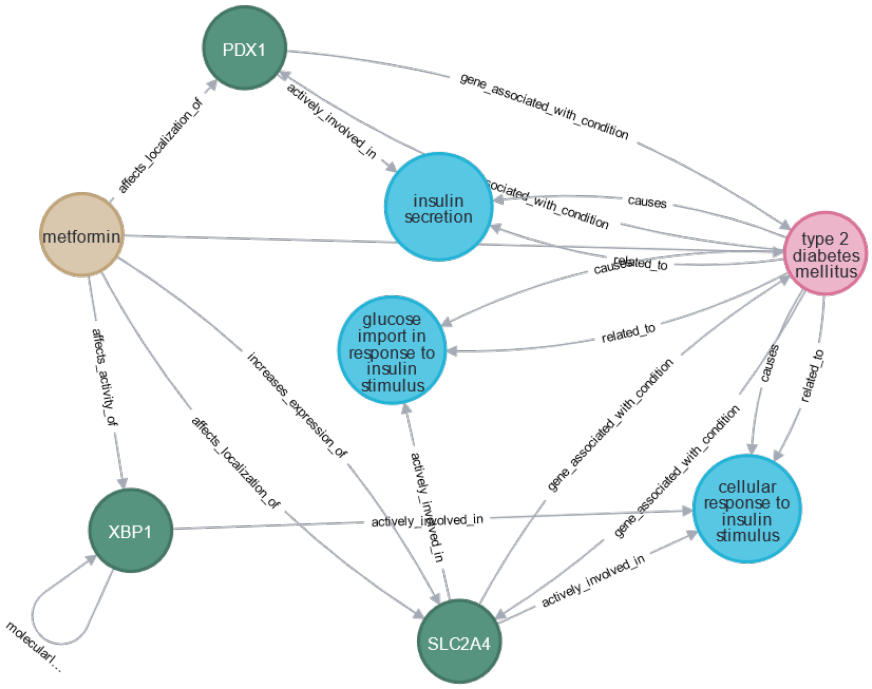


Figure 1.3: An example subset from the Reasoning Over Biomedical Objects linked in Knowledge Oriented Pathways (ROBOKOP) biomedical knowledge graph (Section 1.5.1). This graph shows the interconnected pathways surround the drug metformin. Nodes of brown color represent drug nodes, nodes with green color represent genes, nodes with pink color represent diseases, and nodes with teal color represent biological processes. This graph was generated by the Neo4J interface with the query '**MATCH** (d:disease)–(b:biological_process_or_activity)–(g:gene)–(c:chemical_substance) **WHERE** d.name="*type 2 diabetes mellitus*"AND c.name="*metformin*" **RETURN** *'

enforced support when publishing data. As more interests, such as governmental agencies projects like PrecisionFDA (Olson et al., 2022) and large scale projects have been made to aggregate these data, the demand for extensive standards which cover all potential experiments have become more coveted.

1.3 Hypothesis Generation

The problem of generating novel connections utilizing a database has been previously studied. As applied to knowledge graphs, **hypothesis generation** is the process of finding unknown connections between entities through the automated exploration of a database.(Spangler et al., 2014) Hypothesis generation can take multiple forms, depending on the source dataset and the form of question the user wants answered. If a user is performing genomics studies, it may be as simple

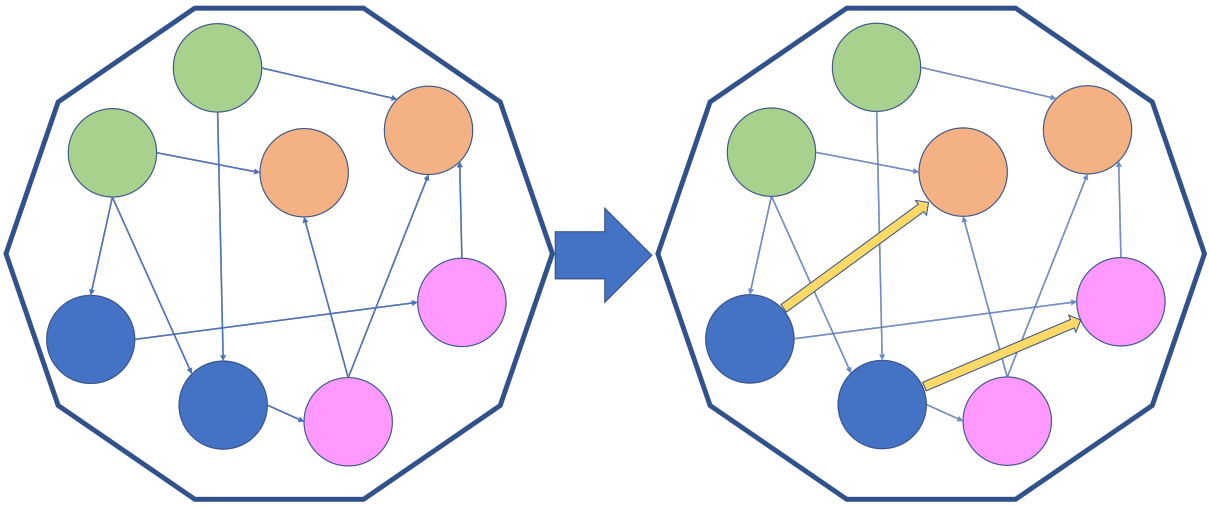


Figure 1.4: A visualization of **hypothesis generation**. In this figure we show how novel edges may be infer on the resultant graph from Figure 1.1. We take as an input an existing knowledge graph. A hypothesis generation process looks to create edges between nodes not yet known to be connected. These newly generated hypothesizes are represented in the right image by the bold yellow arrows.

as a database returning a list of genes. In the space of knowledge graphs, the result of hypothesis generation take the form of pathways of biomedical objects, where each node and edge along the pathways contributes to an overall hypothesis. A user may query with just a source node seeking all possible connections in the graph, or more manageably, they will query with two nodes (a source and tail) and seek to elucidate the connection between two objects.

This exploration is critical to the field of drug discovery and repurposing (Section 1.4); many drugs and diseases are known to be related, but with the underlying biological mechanisms of this interaction are unknown. Hypothesis generation provides us a methodology to explain these unknown mechanisms and relationships, providing domain-experts greater detail to existing drug disease mechanisms.

We present an example visualization of hypothesis generation in Figure 1.4. In this figure we show how an existing knowledge base could be used to generate new potential connections. These connections are visualized with the large yellow arrows, which link hitherto unconnected nodes as candidates for exploration of connections.

1.4 Drug Repurposing

One specific use case in which **hypothesis generation** (Section 1.3) may be of note and use is that of drug repurposing. Drug repurposing is the act of taking a compound known for treatment of one disease and repurposing that compound for the treatment of another disease.(Pushpakom et al., 2019) This provides us with multiple benefits: (1) drugs which are known to treat another disease have often already gone through an extensive FDA approval safety screening process, (2) these drugs are already being made at a large scale which greatly lessens the effort to providing them to patients, and (3) extensive real world studying of mechanisms of actions (MOA) of these drug has been conducted.

Drug repurposing has provided multiple very major advancements in treatments of various real-world diseases. Sildenafil is probably the most famous example of drug repurposing (more commonly known by its brand name Viagra).(Polamreddy and Gattu, 2019) Initially approved as a treatment for hypertension (high blood pressure), sildenafil ultimately was found effective as a treatment of erectile dysfunction.(Boolell et al., 1996) Sildenafil was also repurposed as a treatment of pulmonary arterial hypertension.(Galiè et al., 2005)

Often the diseases which are treated through drug repurposing are rare diseases, these are diseases which affect fewer than 1 in 200,000 people as defined in the United States or 1 in 2,000 as defined in the European Union.(Valdez et al., 2016) Rare diseases by their very nature affect few patients, some of these conditions afflict fewer than a dozen people in the entire world, and they are discussed in much greater detail in Section 4.5.2. It becomes increasingly difficult to conduct meaningful clinical trials and FDA approval on such small patient groups. However, drug repurposing provides these patients access to treatment that may be otherwise impossible to justify, but reusing existing and well-studied drugs. One such example of this is David Fajgenbaum, who was diagnosed with idiopathic Castleman's disease. When he didn't respond well to known treatments, he and his care team were able to repurpose the drug sirolimus to put his Castleman's into remission.

The NIH has created an entire division, the National Institute for Aging, which aims to work on treatments for Alzheimer's Disease and Related Dementia (ADRD). In order to facilitate the discovery of treatments, they created the Drug Repurposing for Effective Alzheimer's Medicines (DREAM) initiative.(Desai et al., 2020; Thambisetty and Aging, 2021) Clinical trials for Alzheimer's Disease have consistently failed.(Mehta et al., 2017) More than 100 compounds have been entered into a government sponsored trial, but all haven't met FDA safety and efficacy standards; the result has been loss of billions of dollars and a lack of tools to combat ADRD. There is a dataset containing longitudinal (over a substantial period of time) clinical data for over 20 million older Americans, containing both their medication and history and their current ADRD prognosis. This shows one of the primary benefits of drug repurposing. When generating hypotheses through computational models, analyzing these historical clinical data allow researchers to use observational validate some drug repurposing hypothesis; finding individuals who have taken already approved drugs in the past and investigating their likelihood of getting Alzheimer's. This methodology is not a substitute for a full clinical trial, but enables faster pruning of unpromising candidates early.

1.5 Specific Biomedical Knowledge Graphs

1.5.1 ROBOKOP

The Renaissance Computing Institute (RENCI) is a collaborative effort between UNC-Chapel Hill, Duke University, and the North Carolina State University. RENCI focuses on developing practical and novel software and infrastructure for solving large problems.(Ahalt, 2017) One of the outcomes of this work is the knowledge graph called Reasoning Over Biomedical Objects linked in Knowledge Oriented Pathways (ROBOKOP).(Morton et al., 2019) ROBOKOP serves as an interface for expert users to query and discover novel information in the field of bioinformatics. ROBOKOP is powered by a large and heterogeneous knowledge graph built from over twenty biomedical data sources, capturing over 9 million distinct biomedical concepts. Using state-of-the-art knowledge graph research was critical in the development of this graph.

Neo4J provides the underlying architecture which ROBOKOP is built on top. The Neo4J graph database software provides many features which enable ROBOKOP to aid biomedical research. One main benefit is the ability of nodes to have multiple labels; such as the drug "minoxidil", which contains the labels named_thing, biological_entity, molecular_entity, and chemical_substance. The flexibility of these labels enables end users to query more broadly when desired, such as asking for any biological_entity which fits a specific parameters, and more narrowly, such as asking for a gene, phenotype, or chemical with certain properties.

Another feature of Neo4J used in ROBOKOP is the ability for arbitrary data to be attached to any node or edge. This enables either a boolean, integer, string, or list to be mapped to keys, similar to relational databases. For minoxidil, one such item is a boolean flag which indicates if the drug has been approved for use by the FDA. These flags enable fast and narrow queries for various pathways in ROBOKOP. One particularly useful case of this is for edges linking nodes, a list of publications which support the edge is included, which enables biomedical researchers to quickly cross reference why an edge is included in a knowledge graph and potentially enable a survey of existing literature into a problem.

We have catalogued and provided a table of the 38 databases underlying the ROBOKOP database in Table 1.1. Each of these databases represents specialized expert knowledge in some domain of bioinformatics, often summarizing thousands of hours of intense research. Additionally, each of these datasets must be curated and standardized to fit an ontology. The breadth and scope of the various datasets integrated into one central repository is notable, combining resources like the Genome-wide association studies (GWAS) (Buniello et al., 2019), which gathers information on the genetics in human with specific diseases, PubChem (Kim et al., 2021; Li et al., 2010) which provides information on over 100 million chemical compounds, and Comparative Toxicogenomics Database (CTD) (Davis et al., 2021; Mattingly et al., 2003) a dataset of toxic compounds and the specific regions of the human body they disrupt.

1.5.2 HetioNet

The HetioNet knowledge graph is an integrated network of twenty-nine different biomedical databases; consisting of genes, diseases, and drugs.(Himmelstein and Baranzini, 2015; Himmelstein et al., 2017) This graph was developed by Himmelstein et al. with the specific intention of being used for drug repurposing tasks (Section 1.4).

The HetioNet knowledge graph is significantly smaller and more focused then the ROBOKOP (Section 1.5.1) graph. HetioNet contains 47 thousand nodes and 2.3 million edges; an order of magnitude less than ROBOKOP (which has 9 million nodes and 255 million edges). The smaller nature of the graph makes it much more focused and less noisy, but also removes many of the unexpected connections which may be found in a larger graph. This makes it a useful secondary dataset for testing novel knowledge graph methodologies.

A catalog with references of the twenty-nine databases which compose the HetioNet graph can be found at `https://git.dhimmel.com/rephetio-manuscript/`.

1.6 Federated Knowledge Graphs

Federated database systems are database which serve as a collection of multiple other databases, with each database run independently of the others.(Risch, 2009) A federated database system take one of two forms; loosely coupled and tightly coupled. A loosely coupled architecture requires the user to interact with all databases within a federated system and maintain knowledge of all component databases. A tightly coupled architecture takes the form of a singular controller database, which a user interacts with; and several peripheral databases which support the controller. In a tightly coupled system, when a user makes a query, the controller manages the query of all constituent elements; this enables many distinct databases to be queried while a user must only maintain knowledge of a singular endpoint.

A promising new branch of research is being developed in Federated Knowledge Graph, that is Knowledge Graphs which access knowledge by querying multiple other databases. These

databases have complex tradeoffs from traditional single database knowledge graphs. A primary downside of a federated approach is speed, queries which used to be fully answerable from a single machine now require multiple API queries to ensure complete accuracy of the solutions. Another issue which becomes more pressing in a federated space is that of ontological choice. Administrators of federated knowledge graphs must be careful that all databases which they interact with are using the same level of ontology; additionally, all peripheral databases must be ensured to be compliant with the ontology of the controller or ontological normalization must be performed by the controller when submitting queries and processing replies.

Presently, at RENCI, work is being done on the creation of federated knowledge graph systems. Strider(Wang, 2021) federated knowledge graph has been created in association with the biomedical translator program. The Strider federated knowledge graph integrates over 33 knowledge provides into a singular access point and ontology. This Strider system will at runtime query each of these 33 providers to find connections between different biomedical objects. This enables the result graph to be as up to date as possible given the current state of knowledge, as long as the API of the queried datasets have been updated, that state will be reflected in the Strider result.

Federated systems do not come without drawbacks. Traditional graph databases gain a great deal of speed and efficiency by having the entirety of its data on a singular system. When transitioning to a federated approach, issues of network latency, query response time, and failed queries start to become prevalent. With these issues, a primary failure of federated systems versus non-federated is speed. Queries can often go from taking tens of seconds to multiple minutes to return a response to an end user.

Because of the speed issues, these databases become much less able to be used in many applications. The need to query and allow all subsequent databases to return a response and dynamically building results is just too onerous for time constraints. Any procedure which can help sort and filter the results of these graphs and restrict them can greatly aid in speeding up their result time.

1.7 Specific Aims

The main goal of my studies is to develop approaches for leveraging specialized domain knowledge when mining knowledge graphs to enable mechanistically valuable and testable hypothesis generation in specialized domains. To achieve this objective, we have the following specific aims.

The first specific aim is ”Leverage properties of knowledge graphs to further knowledge extraction.” In this aim, we will explore how the degree of nodes in a knowledge graph can be utilized for complex hypothesis generation. We focus on issues of node degree, representation of certain nodes in existing hypothesis generation algorithms. Furthermore we explore the creation of novel path ranking algorithms and provide implementations in Neo4J that can help address these problems.

The second specific aim is ”Querying semantically relevant graph patterns for hypothesis generation.” In this aim, we explore the idea of how users interact with the knowledge graph. We introduce the concept of semantically meaningful graph patterns, which are combinations of nodes that provide insight into particular domains. We implement these ideas in the biomedical domain as “Clinical Outcome Pathways”. We further explore how to algorithmically generate Clinical Outcome Pathways for knowledge graphs.

The third specific aim is ”Extending general biomedical knowledge graphs to specific problem domains.” In this aim, we seek to utilize methodologies from Aims One and Two in the subfield of biomedical knowledge graphs. We explore how the biomedical field provides unique computational and engineering challenges and potential solutions to these issues. In addition, we study problem specialized knowledge graphs and knowledge graphs that may be constructed to solve very narrow problems. We introduce two focused biomedical knowledge graph which extend the ROBOKOP knowledge graph; COVID-KOP and SCENT-KOP.

1.8 Existing Approaches

This section describes the existing strategies for hypothesis generation using knowledge graphs. In each of these approaches, we assume we have two objects, or two nodes, which are sought to be explored. These two nodes are then tested in some way, and results are provided to an expert for their connectedness. Existing approaches often treat the graph as irrelevant of context, and fail to incorporate expert users into any form of feedback loop. These approaches can be broadly classified into two categories:

1. Global Optimization Methods

2. Deep Learning Methods

1.8.1 Approach 1: Global Optimization Methods

Global optimization methods are methods which define some function which represents the mathematical optimally way to capture relationships between nodes in a graph (Goyal and Ferrara, 2018). These formulations can be viewed as an optimal way to relate each node to each other in some geometric space (typically by minimizing eucluidan distance in some way). Once properly formulated, the nodes and edges of a graph simply need to be input into this methodology to generate an optimization problem for the graph. Once this problem has been formulated, it can be solved with well studied solution methodologies such as Runga-Kutta (Ausiello et al., 2012).

Methods such as Local Linear Embeddings(Sam T. and Saul, 2000) and Laplacian Eigenmaps(Belkin and Niyogi, 2002) are examples of global optimization methods. These methods operate by attempting to place vector representations of nodes near vectors of all its neighbors. The idea is that by capturing all local clusters of behavior, global behavioral patterns will emerge. By structuring the target embedding as constrained optimization problems, these methods can efficiently and consistently achieve the same results.

Unfortunately, these methods are computationally unfeasible for large graphs. These graphs can have hundreds of millions of nodes and billions of edges. These methods as they do not scale

well in number of edges in a graph (Goyal and Ferrara, 2018). The embedding of every node affects the embedding of each other node, which must be truly calculated to achieve the *global* solution to the problem. These computational restrictions have made these methods untenable for modern knowledge graphs.

1.8.2 Approach 2: Deep Learning Methods

Deep learning has become a common blanket term for a family of approaches to data science and statistical inference problems in the last decade. In this family, large amounts of high quality data is fed into a neural network. The architecture of these neural networks can vary wildly

Graphics compute units (GPUs) are core components of the deep learning approach. Leveraging the rapid advancements in quality, availability, and capability of GPUs has enabled much of the deep learning revolution. GPUs have transformed from niche hardware used to enable high speed graphic calculations for video games to being a cornerstone of high performance computing and deep learning. Recent GPUs have ten times, and even a hundred times, more memory then GPUs of five years ago. Additionally, single GPUs can have several hundred parallel processors. An extreme example is the Nvidia V100 Volta, which has 640 parallel cores.

For almost all forms of deep learning on graphs, the representation of the graph is substantially simplified for the learning process. Fully representing an interconnected graph is too complex, and the vast majority of deep learning processes require inputs of finite size. To accommodate this limitation, random walk samples of the graph are taken. This provides a finite *fingerprint* for the state of the graph. Additionally, given a sufficient amount of samples, random walks should fully capture the transition probabilities for moving from one node to another.

A more practical way to perform machine learning for knowledge graphs is to avoid learning directly on the graph, as described above. The alternative approach is to find a lower dimensional space which we may then use to represent the nodes (and relationships) of the graph. This process is referred to as graph embedding. In the process of graph embedding, we take a standard adjacency matrix representation of the graph and convert all nodes to a vector of some user

specified length s, where s is substantially less then n. As discussed further below, methods provide different properties for how embedding vectors relate to each other.

The new generation of graph embedding methods leveraged work from the Natural Language Processing (NLP) space in capturing semantic meaning from natural language. Initially begun with DeepWalk algorithms (Perozzi et al., 2014), and later extended into the famous Word2Vec algorithms (Grover and Leskovec, 2016), these works were trained on an extremely large corpus of English text; leveraging Skip-Gram Neural Networks to learn a model of language. These works have provided breakthroughs in NLP space by enabling the vectorization of words into a "semantic space" (Grover and Leskovec, 2016). The claimed ability of this "semantic space" was to place words of similar meaning near each other in this high-dimensional space, and words which are dissimilar are embedded far apart. A famous example of the Word2Vec's ability to capture semantic ideas was the following example: when looking at embedded vectors for the words King, Queen, Man, and Woman, it was seen that the following held **King - Man + Woman = Queen** (Drozd et al., 2016). The algorithm was able to capture the internal semantic relationship of how the meaning of King and Queen differ by gender. The ability for a computer to capture higher order semantic relationships from direct observation of text with no interference of human experts is a very powerful result. Although further research has countered that many of these bold claims of the powers semantic embedding may be the results of misreadings of the data or the result of ill-designed experiments (Gonen and Goldberg, 2019; Nissim et al., 2020), these models still provide the backbone of much of the progress in the exploration of computational understanding of how ontological entities relate to each other.

Node2Vec is a popular graph embedding method (Grover and Leskovec, 2016), adapting the work done for Word2Vec into the graph space. Node2Vec randomly walks the input graph, treating each walk through the graph as a unique "sentence". It then attempts to place all nodes in these "sentences" near each other, by extensively training a neural network. This approach avoids many of the complexity issues of the global approaches, by allowing the graph to be discretized as sentences instead of attempting to capture all the relationships. GraphSage builds upon Node2Vec.

GraphSage inductively builds out information about the graph, starting at a few nodes and slowly constructing a more complex understanding of how nodes relate to each other by building out and defining more regions of the graph as it grows.(Hamilton et al., 2017) Additionally, some work has been done on changing the random walks which power graph embedding algorithms. Meta-context Aware Random Walks (MARU) (Jiang et al., 2020).

Translational embeddings serve as a further development of stochastic machine learning models. These models began with TransE, which was published in 2014 by Wang et al (Wang et al., 2014). Further work on this problem has lead to the creation of a whole family of translational graph embedding models. Some examples of this family include: TransA (Xiao et al., 2015), TransR (Lin et al., 2017), TransD (Ji et al., 2015), TranSparse (Ji et al., 2016), MTransE (Chen et al., 2017). The intuition for these methods is in utilizing the node and edge labels of knowledge graphs. The methods treat nodes similarly to Node2Vec, attempting to capture how nodes can be represented in an embedding space by training a neural network. But, simultaneously, to learning node embeddings, it also considers the edges that connect these nodes; the algorithm tries to capture these edges as mathematical transitions between the embedded nodes. Training additional neural networks on the edges of a graph, representing both nodes and edges in one unified space. By capturing the meaning of both nodes and edges at once, these algorithms further understanding of the underlying structures and meaning of the knowledge graph.

Another approach to hypothesis generation from knowledge graphs is to utilize machine learning methods classify nodes and edges. These methods can take the form of various statistical prediction tools, such as regression, support vector machines (SVMs), neural networks. These approaches try to represent and find patterns on the nodes and relationships between nodes. An immediate issue presents itself with this approach, as these graphs can contain millions of nodes and extremely complex relationships between these nodes. Let us define the number of nodes in a graph as n. If we imagine a scenario in which we performed machine learning methods directly on an adjacency matrix for a graph, sometimes referred to as One-Hot encoding, each node would

require n binary values to represent it. That requires our machine learning algorithm to be able to deal with millions of binary input values where the majority of these values will be zero.

Hypothesis generation may utilize graph embedding techniques. The current work for utilizing hypothesis generation is based upon translational embeddings (Wang et al., 2014). These embeddings seek to embed nodes as points in hyperspace, and simultaneously capture edges as mathematical operations which move from one node to another. To generate hypothesis, translational embeddings are generated for a graph. Once these embeddings have been created, the embeddings are systemically searched for two nodes which potentially could be connected by an edge operation, but presently are not linked (Akujuobi et al., 2020; Sosa et al., 2020; Sybrandt et al., 2020). Work leveraging graph embeddings for hypothesis generation has been focused on the space on biomedical knowledge graphs. Automatic graph-mining and transformer based hypothesis generation approach (AGATHA)(Sybrandt et al., 2020) was created to enable faster drug discovery pipelines, it utilized PyTorch-BigGraph (PYBG)(Lerer et al., 2019), finding embeddings for all nodes in a knowledge graph knowledge graph, and then optimizing known mechanistic connections (Sybrandt et al., 2020).

1.9 Scope

The scope of this thesis is concerned with biomedical knowledge graphs. We aim to provide solutions to problems encountered in our exploration of knowledge graphs. All solutions and tools presented in this thesis should be general to all knowledge graphs, but we have verified their efficacy only in biomedical situations.

Our scope does not include the task of constructing an entire knowledge graph, although we do explore the task of constructing specialized variants of biomedical knowledge graphs in Chapter 4.

We give three reasons for focusing primarily on biomedical knowledge graphs.

1. The biomedical field presents the most obvious case for highly interactive ontological data. Many centuries of work have been spent cataloging the human body, it's genome,

diseases, their symptoms, and how those symptoms relate back to the body. In this case, we fortunately have much of the heavy lifting of building ontological classification done for us.

2. The existence of high quality biomedical knowledge graphs provide us a datasets on which to conduct real world experiments on the efficacy of our methods.

3. Our group and collaborators are constructed of many individuals with extensive biomedical expertise. Verifying the significance of particular patterns and pathways requires domain expertise so we are able to leverage the expertise provided to us. These individuals are thanked in the Acknowledgements Section.

1.10 Thesis Statement

As the quality and complexity of knowledge graphs improve, the ability to utilize these graphs for high quality hypothesis generation also improves. We seek to further the hypothesis generative capabilities of knowledge graphs. *Leveraging the inherent mechanistic patterns of biomedical knowledge graphs, we seek to enable the generation of high quality and useful biomedical hypotheses.*

1.11 Dissertation Overview

The remainder of this dissertation is organized into the following chapters.

- **Chapter 2** describes the issues of ranking in knowledge graph pathways. We explore the ranking issue and attempt to tackle it through utilizing the degrees along the pathways.

- **Chapter 3** outlines our conception of Clinical Outcome Pathways (COPs), a specific combination of node and edge labels which are useful for the problem of drug discovery and drug repurposing (Section 1.4). We then explore the generalization of this idea in through our conceptualization of *semantic query pattern* , in which experts attempt to build specific combinations of nodes and edge labels to find meaningful patterns in knowledge.

- **Chapter 4** describes the advancements made in specialized niche-domain knowledge graphs. Producing biomedical knowledge graphs for unique discipline specific problems and how we can leverage these knowledge graphs to uncover hypothesis in these niche cases.
- **Chapter 5** concludes the work and discusses the implications of future progress in the combinations of biomedical and computational research, especially regarding access to care and ontologies.

Table 1.1: A catalogue of the sources ROBOKOP uses to gather their biomedical data. Each of these sources is queried, and must be normalized into various different standard ontology for biomedical objects. The ultimate result of this integration can be viewed at `https://robokop.renci.org/`.

Aeolus (Banda et al., 2016)	AmiGO (Carbon et al., 2009)	Bio2RDF (Belleau et al., 2008)
BioGRID (Oughtred et al., 2021; Stark et al., 2006)	CHEBI (Hastings et al., 2016)	Chem2Bio2RDF (Chen et al., 2010a)
ChEMBL (Mendez et al., 2012, 2019)	ClinGen Allele Registry (Pawliczek et al., 2018)	ClinVar (Landrum et al., 2014)
Comparative Toxicogenomics Database (CTD) (Davis et al., 2021; Mattingly et al., 2003)	DrugCentral (Avram et al., 2021; Ursu et al., 2017)	Drugbank (Wishart et al., 2008)
Ensembl (Howe et al., 2021; Hubbard et al., 2002)	GTEx (Lonsdale et al., 2013)	Genome-wide association studies (GWAS) (Buniello et al., 2019)
Gene Ontology (GO) (Gene Ontology Consortium, 2021)	HUGO Gene Nomenclature Committee (HGNC) (Povey et al., 2001; Tweedie et al., 2021)	Human Metabolome Database (HMDB) (Wishart et al., 2009, 2007)
Human Phenotype Ontology (HPO) (Robinson et al., 2008)	KEGG (Kanehisa et al., 2002)	MeSH (Lipscomb, 2000)
Monarch API (Biolink) (McMurry et al., 2016; Shefchek et al., 2020)	MONDO (Mungall et al., 2017)	Mouse Genome Informatics (MGI) (Bult et al., 2019; Smith et al., 2019)
Mychem.info (Wu et al., 2022)	NCBI Gene (Coordinators, 2016)	Online Mendelian Inheritance in Man (OMIM) (Amberger et al., 2015, 2019)
Orphanet database (Nguengang Wakap et al., 2020)	PANTHER (Mi et al., 2021)	PharmGKB (Whirl-Carrillo et al., 2012, 2021)
Pharos (TCRD) (Nguyen et al., 2017)	PubChem (Kim et al., 2021; Li et al., 2010)	PubMed (Canese and Weis, 2013)
QuickGO (Binns et al., 2009)	UniChem (Chambers et al., 2013)	UniProtKB (Bateman, 2019)
mygene.info (Wu et al., 2013; Xin et al., 2016)	myvariant.info (Xin et al., 2016)	

CHAPTER 2

Promiscuity Path Scores

2.1 Introduction

Research and development processes in drug discovery are notoriously time- and labor-intensive, often requiring billions of dollars for a single compound to reach the market (Pushpakom et al., 2019). As a result, the gap between existing treatments and medical needs has been steadily growing. *Drug repurposing* (also known as *drug repositioning)* is a strategy for exploring new uses for FDA-approved drugs that are beyond the original medical indication for the drugs (Pushpakom et al., 2019; Corsello et al., 2017; Singh et al., 2020; Oprea and Mestres, 2012). This approach has the potential to significantly reduce the costs and risks of discovering new drugs, by enabling new treatments to be generated faster and at lower prices to both the companies and patients (Parvathaneni et al., 2019). The growing ease of access to large-scale biomedical *knowledge graphs (KGs),* e.g., (Morton et al., 2019; Nelson et al., 2019; Himmelstein et al., 2017), has offered researchers major opportunities for developing computational drug-repurposing approaches (Luo et al., 2017; Zhu et al., 2020), which promise to further improve the efficiency of drug repurposing. Notably, rare diseases (Sosa et al., 2019; Zhang et al., 2021) and COVID-19 (Zhang et al., 2021; Zhou et al., 2020; Wang et al., 2020; Zhang et al., 2021; Yan et al., 2021) have been some of the recent popular targets of computational drug repurposing.

The process of drug repurposing can be viewed as consisting of two stages: (1) identification of candidate drugs for a given disease, and (2) clinical trials of the candidates to test their viability. In this thesis we consider computational approaches that apply to the first stage of the process. Our specific interest is in developing approaches for automatic discovery and refinement (via ranking)

of candidate *drug-treats-disease facts* from biomedical KGs. Each such fact is a hypothesis that a given drug can treat the disease of interest. We focus on the problem of reducing the time and labor costs involved in drug repurposing, by developing approaches that automatically (i) discover potentially viable drug-treats-disease hypotheses, and (ii) rank the resulting hypotheses, thus enabling biomedical experts to effectively cut the collection of hypotheses down to a manageable list of the most promising candidates to undergo clinical trials.

The computational approaches to discovering and ranking drug-treats-disease hypotheses that we introduce in this chapter are based on the process of *inference* of new information in KGs. Recall that biomedical KGs are designed to store state-of-the-art biomedical information in computer-tractable ways. Specifically, *biomedical concepts* are represented in KGs as (typed) nodes, while *connections* are represented as named edges between the nodes, see, e.g., (Bizon et al., 2019). Two sources of information in KGs about biomedical concepts and connections are (1) publicly accessible biomedical databases that provide high-quality expert-curated information, such as DrugCentral (Ursu et al., 2016) or Pharos (Nguyen et al., 2017), and (2) text mining on biomedical publications (Lee et al., 2020). As a simple example, two KG nodes, `penicillin` of type `drug` and `infection` of type `disease`, taken together with a directed `treats` edge between the nodes, form the *KG triple* (`penicillin`, `treats`, `infection`). This KG triple captures the information that penicillin treats infections (Fleming, 1929).

Inference of new information from a given biomedical KG works via derivation of *latent connections* between nodes in the KG, that is, of knowledge that is justified by the information in the KG and can be represented as a graph edge, but is not already explicitly present in the graph. This can be viewed as the problem of hypothesis generation (Section 1.3). The approaches that we introduce focus on inference of latent `treats` edges between drug and disease nodes in KGs. As discussed above, such inference can result in potentially viable hypotheses that could be further analyzed and tested by experts for drug repurposing.

Exploration of latent connections in (originally) knowledge bases and (later) KGs goes back decades, all the way to Swanson (Swanson, 1986) using in 1986 a network of research-paper citations

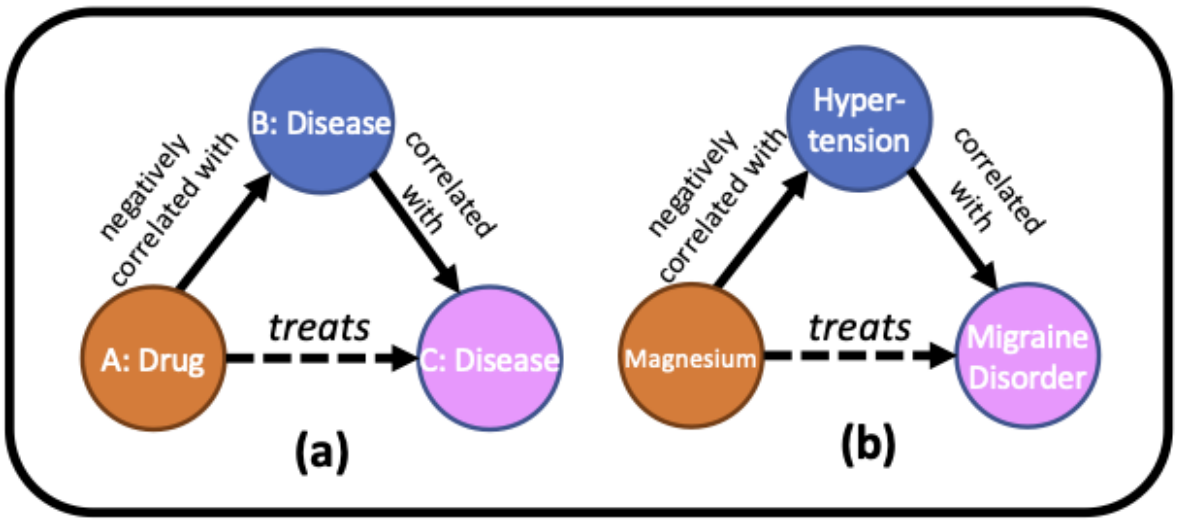


Figure 2.1: The computational approaches to discovering and ranking drug-treats-disease hypotheses that we introduce in this chapter are based on the process of *inference* of new information in KGs. Recall that biomedical KGs are designed to store state-of-the-art biomedical information in computer-tractable ways. Specifically, *biomedical concepts* are represented in KGs as (typed) nodes, while *connections* are represented as named edges between the nodes, see, e.g., (Bizon et al., 2019). Two sources of information in KGs about biomedical concepts and connections are (1) publicly accessible biomedical databases that provide high-quality expert-curated information, such as DrugCentral (Ursu et al., 2016) or Pharos (Nguyen et al., 2017), and (2) text mining on biomedical publications (Lee et al., 2020). As a simple example, two KG nodes, `penicillin` of type `drug` and `infection` of type `disease`, taken together with a directed `treats` edge between the nodes, form the *KG triple* (`penicillin`, `treats`, `infection`). This KG triple captures the information that penicillin treats infections (Fleming, 1929).

to find underexplored relationships between biomedical entities. The methodology of "Swanson's ABC triangle" (Baker and Hemminger, 2010), see Figure 2.1 for an illustration, generalizes the approach of (Swanson, 1986) into a well-accepted inference technique. The methodology applies to triangle-shaped subgraphs of biomedical KGs with concept nodes `A`, `B`, and `C` and edges `ab` from `A` to `B` and `bc` from `B` to `C`, with a missing edge between `A` and `C`. (Recall that such an "ABC triangle" would be represented in the KG via two KG triples, (`A`, `ab`, `B`) and (`B`, `bc`, `C`).) The objective of the inference is to use such "ABC triangles" to infer in the KG an edge relationship, `ac`, between the nodes `A` and `C`, where `ac` would be *justified* by the path from `A` to `C` (via `B`) through the edges `ab` and `bc` that form the triangle subgraph, provided certain conditions hold on the node and edge names on the path. (We will turn to a discussion of the required conditions shortly.) In

case of success, the inference would add to the KG an *inferred KG connection* represented by the KG triple (A, ac, C).

Figure 2.1 presents an illustration. Part (a) of the figure shows a particular *pattern* of Swanson's ABC triangle (Swanson, 1986; Baker and Hemminger, 2010): The triangle pattern requires node A (node B, node C, respectively) to be of type drug (of type disease, of type disease, respectively), and requires the edge ab (edge bc, respectively) to be named negatively correlated with (correlated with, respectively). In a given biomedical KG, this pattern would apply to those subgraphs of the KG whose node types and edge names match those specified by the nodes and solid edges in the pattern. Figure 2.1(b) shows one such specific instance, in the ROBOKOP KG (Morton et al., 2019), of the graph pattern of Figure 2.1(a), as follows: The node of type drug in the pattern is instantiated with magnesium, first node of type disease – with hypertension, and second node of type disease – with migraine disorder. The solid edges going from magnesium through hypertension to migraine disorder have the same names in the graph pattern in Figure 2.1(a) and in the pattern instance in Figure 2.1(b). As argued in (Baker and Hemminger, 2010), the existence of such a pattern instance in a biomedical KG would warrant the derivation of the inferred treats connection between magnesium and migraine disorder, shown in Figure 2.1 via dashed edges. The inferred treats edge would justify biomedical experts in studying further whether magnesium can be repurposed to treat migraine disorder.

As mentioned above, inference of a latent connection in a biomedical KG would succeed if the path presenting the inference opportunity satisfies certain conditions. Conditions of interest to us in this chapter can be summarized as the path *mechanistically justifying* the inference of the latent connection, that is, the path representing the biological processes involved with the interactions between the source and target entities of the path. Consider, for instance, the generic path pattern that is shown in solid lines between the drug source and the disease target in Figure 2.1(a); it shows a drug-disease connection via another disease that is negatively correlated with the drug and correlated with the disease target. Suppose that biomedical experts believe that this path pattern is rooted

in real-life biomedical processes and is thus worth considering as a justification when deciding whether the given drug treats the given disease. In this case, we say that in each specific instance in a given KG of this path pattern, see, e.g., the solid-line path in Figure 2.1(b), the inference of the `treats` edge, shown via a dashed line in the figure, is *mechanistically justified* by the solid-lines path, due to it being an instance of the expert-approved path pattern.

In this chapter we are interested in working with paths that mechanistically justify inference of the `treats` edges between the nodes for the drugs and diseases of interest in biomedical KGs. Indeed, the existence of each such path enables us to infer the `drug treats disease` candidate hypothesis for the drug-disease pairs of interest. As discussed above, these hypotheses can then be offered to biomedical experts for further selection by hand of the most promising drug-repurposing candidates to undergo clinical trials. The computational task of discovering `drug treats disease` candidate hypotheses from biomedical KGs would be trivial if all path patterns that can mechanistically justify inference of the `treats` edge between drugs and diseases of interest were known. (If this were the case, the computational hypothesis-ranking task would not be a problem either, provided that biomedical experts could rank the known paths, ahead of time, from the most trustworthy to the least.)

It is true that some path patterns that could mechanistically justify inference of connections in biomedical KGs have been described in the literature, see, e.g., (Baker and Hemminger, 2010; Capuzzi et al., 2018). At the same time, formal studies of such path patterns have just begun (see Section 3.2). This means that at this point, not all path patterns that have the potential to lead to viable `drug-treats-disease` hypotheses have been discovered. We thus focus our efforts on the **research objective** of effectively and efficiently finding ways to computationally discover *new* path patterns that could be helpful in computational drug repurposing. To the best of our knowledge, this research objective has not been considered in the literature. Further, achieving it is not straightforward. Indeed, a brute-force approach to discovering new useful path patterns would be to (i) discover all possible paths in a given biomedical KG between drugs and diseases of interest, and then (ii) present the resulting paths to biomedical experts, so that they could choose paths that

can mechanically justify the derivation of the `treats` connection between the drugs and diseases. Unfortunately, this straightforward approach does not scale, due to the number of candidate paths that can be discovered being, in the worst case, exponential in the number of nodes and edges even in the immediate vicinity of the drugs and diseases of interest. (E.g., the total number of paths with at most three edges that could be discovered from the ROBOKOP KG for the magnesium-migraine combination is 20,842.) Clearly, in general it is likely to be time- and labor-intensive for biomedical experts to study all the candidate paths that arise for a given drug-disease pair from the given biomedical KG.

To address the computational complexity of the problem of automatically discovering new path patterns that could be helpful in drug repurposing, we propose to focus on KG nodes called *hubs.* Hubs, a known phenomenon in KGs and other networks (Newman, 2010), are defined as nodes that have an exceptionally high *degree* (defined as the number of adjacent edges) compared to the other graph nodes. Examples include celebrities on social media networks (Lim and Datta, 2012) and "central authority" web sites on the World Wide Web (Kleinberg, 1999). We will be referring to hub nodes in biomedical KGs as *promiscuous nodes;* this terminology has arisen from the notion of promiscuous drugs (Overington et al., 2006), that is, drugs known to interact with multiple biological targets. The promiscuous-node phenomenon in biomedical KGs reflects real-world research, in which certain concepts, e.g., diabetes, become extensively studied and, in some cases, perhaps even overstudied, thus leading to an abundance in the KG of connections of various kinds to these nodes, while other more complex or less promising concepts could remain underexplored (Resnik, 2001).

Suppose a biomedical researcher aims to understand the connection between drug `atorvastatin` and `heart disease` (as seen in Figure 2.2). Recall that the real-world drug-disease connection that researchers seek to uncover is *mechanistic.* Thus, in their evaluation of the drug-disease paths returned from the KG the researchers are likely to prioritize pathways through biomedical entities that deal directly with the biological process involved with the interactions between the entities (Dwyer et al., 2005; Röhl, 2012; Sand, 2018). As suggested by the experimental results reported in

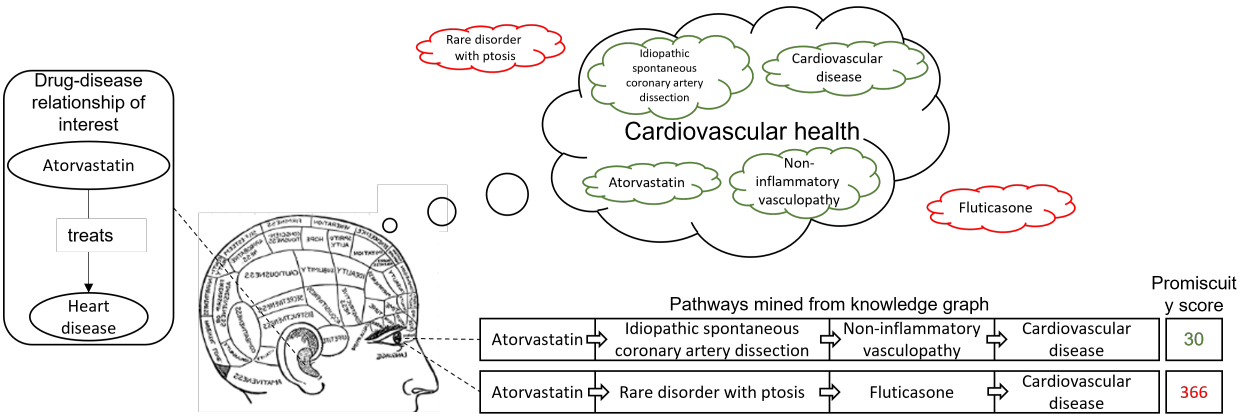


Figure 2.2: A visualization of potential differentiation in a researcher's mind between various knowledge-graph pathways for a drug-disease relationship of interest. The drug-disease pair can induce a "space of associations" in the researcher's mind, creating a preconceived reasoning framework. When viewing pathways discovered between the drug and disease nodes in a knowledge graph, the researcher can prioritize pathways of interest based on whether the intermediate nodes in the pathway fit into the preconceived framework. The promiscuity values and scores introduced in this chapter can help the researcher by automatically prioritizing those pathways whose intermediate nodes potentially fit into the relevant research context.

this chapter, such *mechanistically justified* pathways often go through intermediate KG nodes in the immediate drug-disease neighborhood that do not have excessive linkage with the rest of the KG; that is, these intermediate nodes are unlikely to be promiscuous.

As it turns out, promiscuous nodes can complicate the inference process. Indeed, they often dominate the results of KG queries and can also become overly central to embedding methodologies (Cai et al., 2018; Goyal and Ferrara, 2018; Wang et al., 2017). This effect may cause difficulties for analysts in their efforts to find meaningful paths in a biomedical KG between nodes of interest (e.g., from a drug node to a disease node), as well as in implementing graph embeddings (Jain et al., 2021). Based on these observations, in our proposed algorithms for automatically discovering new path patterns that could be helpful in drug repurposing we focus on prioritizing paths that do *not* traverse promiscuous nodes. We believe that this approach follows the intuition for how biomedical experts evaluate pathways discovered by KG analysis. Consider Figure 2.2, in which we sketch our understanding of the possible mental model. Suppose a biomedical researcher aims to understand the connection between drug `atorvastatin` and `heart disease`. Recall that the

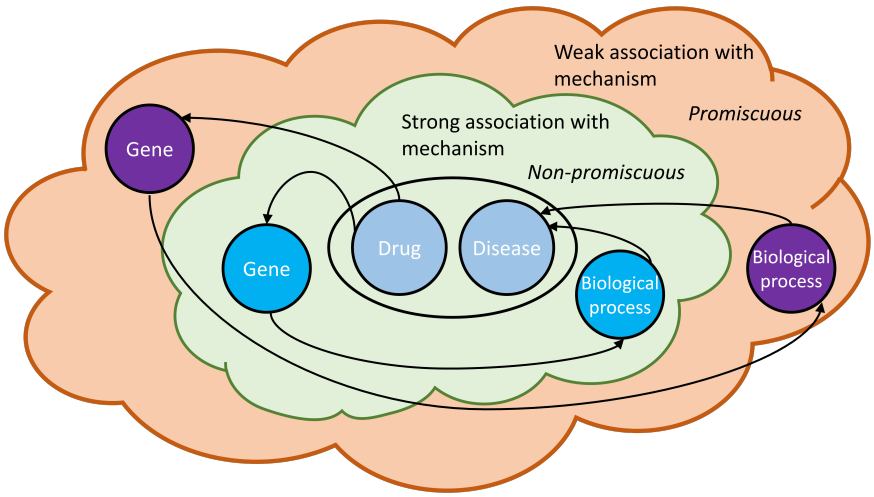


Figure 2.3: A visualization of the potential differentiation between various knowledge-graph pathways for a drug-disease pair of interest shown in the center of the figure. In the inner (green) "association cloud," we show nodes with strong associations with the drug-disease mechanism of action. The drug-disease pathways that stay in this inner cloud could be of interest to biomedical experts. In the outer (orange) "association cloud," we show nodes with weaker connections to the drug-disease mechanism of action. The drug-disease pathways that traverse this outer cloud could be of less interest to the experts.

real-world drug-disease connection that researchers seek to uncover is *mechanistic*. Thus, in their evaluation of the drug-disease paths returned from the KG the researchers are likely to prioritize pathways through biomedical entities that deal directly with the biological process involved with the interactions between the entities (Dwyer et al., 2005; Röhl, 2012; Sand, 2018). As suggested by the experimental results reported in this chapter, such *mechanistically justified* pathways often go through intermediate KG nodes in the immediate drug-disease neighborhood that do not have excessive linkage with the rest of the KG; that is, these intermediate nodes are unlikely to be promiscuous.

This intuition is further summarized in Figure 2.3. In the core of the figure, we have a drug-disease pair that a biomedical researcher may be interested in exploring. The inner cloud around the core contains nodes that have a strong association with mechanisms that relate the drug to the disease; we label drug-disease pathways through these nodes as *useful* because of their close mechanistic relationship to the drug and disease of interest. In contrast, in the outer cloud around the drug-disease pair we find nodes that have a weaker association with the mechanisms that relate

the drug to the disease. Promiscuous nodes will often fall into this category. We refer to pathways that go through this outer cloud as *noisy,* as they are less relevant to the mechanistic drug-disease relationship.

Building on the intuition sketched in Figures 2.2 and 2.3, we have developed and implemented algorithms that automatically discover promiscuous-node-avoiding paths between the given drug and disease nodes in a given biomedical KG. Our algorithms are general and domain independent (that is, they can be applied to tasks other than drug repurposing), and have the advantage of not having to use any user-specified languages or external help from, e.g., embedding models. Instead, they exploit existing node properties in the given knowledge graph. Our algorithms also automatically rank the discovered paths by the maximal degree of their intermediate nodes, which we refer to as *promiscuity score.* The resulting ranked lists can be offered to biomedical experts, as aids in helping them to decide on the mechanism of action of the drugs on the diseases in question. The experts' decisions can further influence the downstream drug-repurposing workflow, with potentially clinical studies scheduled for the most promising drugs based on the quality of the mechanistic justifications provided by the discovered drug-disease pathways.

Our **specific contributions,** as presented in this chapter, are as follows.

- To achieve the research objective of effective and efficient computational discovery of novel path patterns in biomedical KGs that could be helpful for drug repurposing, we propose to focus on paths in KGs that avoid traversing promiscuous graph nodes;
- We introduce path-discovering algorithms that effectively and efficiently prioritize paths that avoid traversing promiscuous nodes in the given biomedical KG, thus increasing the likelihood of discovering paths that mechanistically justify the drug-treats-disease connection between the given drugs and diseases;
- We report the results of a case study for comparing multiple KG-derived paths between the drugs of interest and the diseases that the drugs could treat; our observations suggest that

the proposed algorithms have the potential to solve the challenges of promiscuous nodes in computational drug repurposing with KGs;

- We report results for differentiating between positive and negative drug-disease pairs, i.e., those in which the drug could treat the disease and those in which it could not, respectively; the results suggest that the proposed algorithms show promise in differentiating between positive and negative drug-disease pairs; and

- We report results for classifying positive and negative drug-disease pairs by a classifier trained with promiscuity scores introduced in this chapter; our findings show that the classifier has the potential to successfully distinguish positive and negative drug-disease pairs.

To summarize, our findings suggest that the general domain-independent algorithmic approaches introduced in this chapter have the potential to address effectiveness and efficiency challenges in task-guided computational path discovery, in the biomedical domain and conceivably beyond.

Chapter outline. We discuss related work in Section 2.1.1. Section 2.2 presents the preliminaries and our research tasks and hypotheses. Section 2.3 introduces our proposed approaches. We discuss the case study in Section 2.4. Section 2.5 details the implementation of the proposed approach and the results of the experiments. In Section 2.6 we discuss the limitations of our methods. Finally, Section 2.7 concludes the chapter.

2.1.1 Related Work

The scale-free network model of (Barabási and Albert, 1999) helps explain why promiscuous nodes may be seen in knowledge graphs *(KGs).* The model states that the distribution of edges in a network can be described by the relationship $P(k) \sim k^{\gamma}$; here, $P(k)$ is the probability that a randomly sampled node has degree k, with fixed-value $\gamma \in (2, 3)$. According to Barabási and Albert (Barabási and Albert, 1999), scale-free graphs arise when two conditions are satisfied

1. *Growth*: in which a graph may gain more nodes over time, and

2. *Preferential attachment*: by which nodes that already have many edges will gain more edges.

Although there has been some controversy as to whether scale-free networks are common in certain disciplines, they appear to be prevalent in the biomedical domain (Broido and Clauset, 2019). Indeed, biomedical KGs reflect biomedical research, which satisfies both conditions of (Barabási and Albert, 1999). Specifically, biomedical research demonstrates *growth*: As more research is conducted, more ideas and concepts are discovered and introduced in the biomedical field (Doğan et al., 2021). Further, biomedical research exhibits *preferential attachment*: as certain key concepts tend to be explored more readily than others (Resnik, 2001).

Heuristic methods for finding mechanistically justified pathways between drug and disease nodes in KGs, such as the Swanson's ABC triangle (Swanson, 1986; Baker and Hemminger, 2010) discussed in Section 2.1, traditionally rely on user insight for hypothesis generation. Being labor intensive, such approaches cannot scale to massive biomedical KGs. In this chapter, we focus on discovering more robust and scalable approaches for domain analysts to find meaningful pathways, in drug repurposing and possibly beyond.

Research into mechanisms of drug action is a long-standing area of study for pharmacists. Traditional methods for representing how drugs treat diseases, such as drug targets and mechanisms of action, have given rise to better understanding of some drug-disease relationships. Biomedical KGs can be used to scale up and speed up discovery of such mechanisms. As an example, the study of Clinical Outcome Pathways (COP) (Capuzzi et al., 2018; Korn et al., 2022) for drug-disease pairs is an effort to robustly define how a drug and disease relate through a series of KG nodes and edges. Consider, for instance, a Swanson's ABC triangle (Swanson, 1986; Baker and Hemminger, 2010) in which node A stands for `drug`, node B stands for `target`, and node C stands for `disease`. The meaning of such a COP is that the drug might treat the disease by acting on certain intermediate target nodes linked to the disease (Capuzzi et al., 2018). Starting at the point at which a patient takes the drug, such a COP represents the chain of bio-molecular events as a series of nodes and

edges; this chain is expected to ultimately lead to the resolution of the disease state. COP pathways can provide drug researchers with insights into how drugs and diseases interact, and could be very useful for finding novel drug-disease connections for drug repurposing. The concept of promiscuity score that we introduce in this chapter may prove useful as a metric for ranking COPs generated with biomedical KGs. For more information on COPs please see Section 3.2.

To efficiently find meaningful pathways in biomedical KGs, we explore methods that incorporate path-scoring metrics. Path-ranking *(PR)* algorithms, such as those of (Mazumder and Liu, 2017; Lao et al., 2011), have been proposed to solve KG-completion problems. The main idea of using PR algorithm to complete a KG is in finding paths between two entities in the KG and in then using those paths as features to train a model for predicting missing relations between the nodes. Given an entity-pair (s, t), a typical PR-based method performs random walks over the given KG to find paths with fixed length starting from s and ending at t (Lao et al., 2011; Gardner et al., 2013, 2014; Li et al., 2014). In addition, Mazumder and Liu (2017) utilized the Word2vec model (Mikolov et al., 2013) for contextual feature training, proposing context-aware PR (Mazumder and Liu, 2017) to find paths. Observe that using random walks or embedding methods for finding paths would be affected by promiscuous nodes. Indeed, random walks naturally gravitate towards nodes of high degree and away from nodes with low connectivity. In our research project, we focus on developing more robust approaches to finding meaningful pathways in biomedical KGs.

Other scoring metrics for paths in biomedical KGs have also been developed, see, e.g., (Thafar et al., 2020; Bordes et al., 2013; Zhang et al., 2021). Such metrics typically rely on external sources or embeddings to get path scores. Consider DTiGEMS (Thafar et al., 2020), a method that relies on specific interactions between node types to develop weights for connections; these weights are then multiplied together to provide a final score. Other methods, such as Translational Embeddings (TranSE) (Bordes et al., 2013), rely on embedding nodes and edges of the knowledge graph, and leveraging the embeddings to find how likely an undiscovered connection is to occur. Zhang et al. (Zhang et al., 2021) developed a path-scoring system that utilizes cosine-similarity values from Word2Vec (Mikolov et al., 2013) models. This path-scoring mechanism relies on

word-embedding models built over the names and identifiers of nodes in the graph. This approach limits the use case to words that are known and commonly used in data sets that a language model may ingest.

Drug repurposing with KGs can be performed via inferring latent connections from pathways in KGs (Sosa et al., 2019). As previously discussed, we choose to not use graph-embedding tools for finding KG pathways that mechanically justify the latent `drug-treats-disease` connections. This is due not only to the dependence of the quality of embeddings on data-set characteristics (Jain et al., 2021), but also to the computational time and memory required to digest entire KGs in the deployment of embedding tools, see, e.g., (Hou et al., 2022). In particular, such approaches are not practical for domain experts when studying a specific pathway.

Work by Doğan et al focuses on mining the CROssBAR biomedical network (Doğan et al., 2021), which has been created to infer biological mechanisms from the available biomedical knowledge. The CROssBAR KG gets enhanced by adding edges inferred using a methodology based on machine learning. In their study, the authors of (Doğan et al., 2021) differentially weigh nodes with high degree, to which they refer as "hub nodes," – another example of a research team finding KG nodes of high degree to be an issue in their analysis.

The work of Binder et al (Binder et al., 2022) presents another example of biomedical researchers encountering an issue with node promiscuity in KGs. In their project, the authors of (Binder et al., 2022) sought to use machine learning on a protein KG with the aim of identifying genes in which point mutations are associated with Alzheimer's Disease. The approach taken was to vectorize pathways in the KG of choice, and to feed the resulting vectors into a GraphML model optimized with XGBoost. The authors of (Binder et al., 2022) created a custom variable DWPC (Degree Weighted Path Counts) for each gene. This metric is multiplicative in the inverse of the degree of each node, in an effort to dampen the effects of promiscuous nodes on the model. This is an example of machine-learning approaches accounting for node promiscuity when training predictive models.

Curation of data sets, including purging of irrelevant and inaccurate data from databases, is a studied field commonly referred to as *data cleaning* (Ganti, 2009). The term "KG refinement" (Paulheim, 2017) refers to the task of locating and removing errors from already existing KGs. In this current project, we do not attempt to modify any information in the given KGs with our proposed approaches. Instead, we work with the given KGs without changing them, and organize the pathways output by our approaches in terms of utility to biomedical experts. In searching meaningful pathways in KGs, the proposed approaches base their actions on information about node degrees. To the best of our knowledge, using node degrees as a scoring metric for finding pathways in the way presented has not been proposed before.

2.2 Problem Statement

In this section we review the preliminaries and describe our research tasks, hypotheses, and study objectives.

2.2.1 Preliminaries

To formalize the problem of *knowledge-graph (KG)* path searching, we first define knowledge graphs, as follows:

Definition 2.1. Knowledge Graph. A *knowledge graph (KG)* G is defined as a tuple $G = (V, E, T_V, T_E, \zeta, \xi)$, in which: (i) V is the set of nodes $v_1, v_2, ..., v_n$, with each node representing a data entity (e.g., a specific drug or disease); (ii) E is the set of (directed or undirected) edges $e_1, e_2, ..., e_m$, with each edge representing a relationship between two data entities represented by nodes in V (e.g., a drug `treats` a disease); (iii) T_V is the set of predefined node types (e.g., `drug` or `disease`); (iv) T_E is the set of predefined edge types (e.g., `treats`); (v) ζ is a node type mapping function $\zeta : V \rightarrow T_V$; and (vi) ξ is a link (edge) type mapping function $\xi : E \rightarrow T_E$. The *degree* of a node v in a KG G, denoted $degree(v)$, is the number of the edges that are adjacent to v in G.

A KG in which $|T_V| + |T_E| > 2$ is called a *heterogeneous graph,* while a KG with $|T_V| = 1$ and $|T_E| = 1$ is referred to as *homogeneous.* In our study we focus on heterogeneous KGs.

We define paths (pathways) as follows:

Definition 2.2. Path. A *path* $p_{(s,t,l)} = (s, v_1, v_2, \ldots, v_{l-1}, t)$ *of length* l $(l \geq 1)$ in a KG $G = (V, E, T_V, T_E, \zeta, \xi)$ is defined as a sequence of nodes that starts at a *source node* $s \in V$, ends at a *target node* $t \in V$, and passes through $l - 1$ *intermediate nodes* $v_1, \ldots, v_{l-1}$ in V, such that (s, v_1) (v_1, v_2), $\ldots$, (v_{l-1}, t) all are edges in E.

In this chapter we consider the task of computational *path discovery* between the specified source (s) and target (t) nodes in a given KG G. The discovered paths should satisfy certain computational criteria, detailed next, to be considered as viable candidates for *mechanistically justifiable* paths between the given s and t. Once they are ranked based on their computed viability, the candidate paths are presented to biomedical experts; the experts make the final determination on whether each path is mechanistically justifiable for the given source and target.

2.2.2 The Objective Functions

For a specified pair (s, t) of source (s) and target (t) nodes in a given KG G, we are interested in automatically discovering paths from s to t in G such that all the intermediate nodes on each path are low-promiscuity nodes in G. To formalize this objective, we specify *promiscuity scoring* as follows.

Given nodes s and t in a KG G, consider a path $p_{(s,t,l)} = (s, v_1, v_2, \ldots, v_k, t)$ from s to t of length l in G. The *promiscuity value* of the path $p_{(s,t,l)}$ is the maximum degree of all the intermediate nodes $v_1, \ldots, v_k$ in $p_{(s,t,l)}$:

$$\psi(p_{(s,t,l)}) = max(degree(v_1), degree(v_2), ..., degree(v_k)). \tag{2.1}$$

Now consider all possible paths $p_{(s,t,k)}$ of length k from node s to node t in a KG G. We define the *l-promiscuity score* $\Phi_G(s, t, k)$ of the node pair (s, t) in G as the minimum of the

promiscuity values of all these paths:

$$\Phi_G(s,t,k) = \min_{p_{(s,t,k)} \in G} \psi(p_{(s,t,k)}). \tag{2.2}$$

In the case in which there are no paths of length k from s to t, we define $\Phi_G(s,t,k)$ to be positive infinity $+\infty$.

2.2.3 The Research Goal, Research Tasks, and Hypotheses

Our guiding assumption in this study is that *KG paths with lower promiscuity values tend to be more valuable in helping biomedical experts determine the meaning of the relationship between the entities represented by the source and target nodes on the paths.* Thus, given a source node s and a target node t in a KG G, as well as a path length k of interest, our **overall research goal** is to develop and test effective and efficient computational approaches for automatic discovery of paths $p_{(s,t,k)}$ of length k between s and t such that the promiscuity values of these paths, see Eq. (2.1), are as close to the l-promiscuity score $\Phi_G(s,t,k)$ for s and t, see Eq. (2.2), as possible. In the simplest case, our objective is to discover all the paths of length k from s to t whose promiscuity value is exactly $\Phi_G(s,t,k)$. (By the definition given in Eq. (2.2), at least one such path always exists unless $\Phi_G(s,t,k) = +\infty$.) In general, we are interested in discovering up to a specified number, l, of paths of length k from s to t, such that the promiscuity value of each path is within the top k minimal-value positions among the the promiscuity values of all possible paths of length k from s to t in G. That is, we would like to discover up to l paths of length k from s to t such that all the *other* paths of length k from s to t have promiscuity values at least as high as the promiscuity values of the discovered paths. (In the special case in which the total number of possible paths of length k from s to t is lower than l but $\Phi_G(s,t,k) \neq +\infty$, we are interested in returning all the possible paths.)

Our proposed computational approaches for automatic discovery of low-promiscuity paths between KG nodes of interest are presented in Section 2.3. To test these computational approaches,

we address in Sections 2.4–2.5 the following research tasks, by validating their associated hypotheses:

- Our **research task 1** is to determine, for fixed-length paths from disease nodes to drug nodes in biomedical KGs, whether the intermediate nodes in the paths with low promiscuity values *(low-promiscuity paths)* are more associated with the drugs treating the diseases than the paths with high promiscuity values *(high-promiscuity paths).* The associated **hypothesis 1** is as follows: For a given drug-disease node pair, focusing on low-promiscuity KG paths between the drug and disease nodes tends to produce more meaningful (i.e., more supported by scientific evidence) paths than focusing on high-promiscuity paths. In Section 2.4 we provide a qualitative analysis by a domain expert of both low- and high-promiscuity paths for two use cases; the analysis validates by domain-expert verification the hypothesis for these use cases.

- **Research task 2** is to determine whether the l-promiscuity scores $\Phi_G(s, t, l)$ for positive node-pairs in biomedical KGs tend to be significantly lower than the promiscuity scores for negative node-pairs. We define *positive node-pairs (entity-pairs)* as pairs in which the entities for the source and target nodes are meaningfully connected by existing scientific evidence, and *negative node-pairs* as pairs whose source and target nodes are not associated with each other in this sense. (For example, in presence of scientific evidence that a drug can treat a disease we will label the drug-disease node-pair as "positive.") Our **hypothesis 2** is as follows: (1) The values of l-promiscuity scores tend to be significantly different between the given positive and negative node-pairs; and (2) the mean l-promiscuity score for a given set of positive node-pairs tends to be lower than the mean l-promiscuity score for a given set of negative node-pairs. In Section 2.5 we report the resuls of experiments that verify this hypothesis for ground-truth sets of node-pairs via mean l-promiscuity scores for the sets and p-values obtained from statistical tests.

- **Research task 3** is to determine whether a classifier trained by l-promiscuity scores $\Phi_G(s, t, l)$ in biomedical KGs could correctly predict whether a given node-pair is positive or negative. Accordingly, **hypothesis 3** is as follows: Given l-promiscuity scores for testing node-pairs in a biomedical KG and the ground-truth (positive or negative) labels for the pairs, a classifier trained on a training set of positive and negative node-pairs in the KG tends to correctly predict whether the testing node-pairs are positive or negative. In Section 2.5 we report the resuls of experiments that verify this hypothesis for a binary random forest classifier (Ho, 1995) via the precision, recall, accuracy, and F1-score metrics.

- Finally, our **research task 4** is to determine whether l-promiscuity scores $\Phi_G(s, t, l)$ in biomedical KGs can be used as a pruning methodology in evaluating queries in federated (i.e., non-centralized) biomedical KGs. The associated **hypothesis 4** is as follows: When comparing pruning approaches based on k-promiscuity scores to simple alternative pruning methods, the pruning effectiveness will be higher in the scenario using k-promiscuity scores, while the computational overhead involved will not be significant. In Section 2.5 we report the results of experiments with the Strider federated KG[1] that support this hypothesis.

2.3 Algorithms for Finding Paths and Computing Promiscuity Scores

In this section we introduce two computational approaches for finding paths and calculating promiscuity scores (as defined in Section 2.2.2) for pairs of nodes of interest in a given knowledge graph *(KG).* Given a KG G, source and target nodes s and t in G, a desired path length l, and a desired number of paths k, each of the two algorithms returns the l-promiscuity score $\Phi_G(s, t, l)$ along with the top-k least promiscuous paths from s to t in G.

1. https://github.com/ranking-agent/strider

2.3.1 The Naïve Algorithm

The first of the proposed approaches, which we call *the naïve algorithm,* explores all paths between the source and target nodes s and t by walking the KG G in the depth-first manner. That is, we perform a depth-first search *(DFS)* (Tarjan, 1972) of the graph G by following each path from s to t of length l. As the naïve algorithm builds each path of length l from s to t in G, it keeps track of the highest node degree seen along the path as the current proxy for the promiscuity value of the path. (See Section 2.2.2 for the definition.) Once a path from the source node s has reached the target node t, both the path and its promiscuity value are recorded, as long as the path is among the k least promiscuous paths (i.e., paths with the lowest promiscuity values) from s to t seen thus far in G. This way the algorithm is guaranteed to output the k least promiscuous paths[2] in the increasing promiscuity-value order after all the paths from s to t have been explored. The top-ranked path returned by the algorithm is the path whose promiscuity value is the l-promiscuity score $\Phi_G(s, t, l)$.

In the search for paths of length l from s to t in the KG G, each node n of G gets assigned three attributes: $n.depth$, $n.score$, and $n.parent$. The value of $n.depth$ represents the depth at which the node lies in the current path p, i.e., the first non-source node in p has depth 1, the second has depth 2, etc. The value of $n.score$ is the maximum degree of all the nodes along the path p from s to n. In other words, it is the promiscuity value of the portion of p between s and n. Finally, $n.parent$ stores the node immediately preceding n along the path p. Once a complete path $p_{(s,t,l)}$ from s to t is found, its promiscuity value $\psi(p_{(s,t,l)})$ is stored in the attribute $p.score$.

We present the pseudocode for the naïve approach in Algorithm 1. As mentioned, the algorithm takes in five inputs: the graph G, the source node s, the target node t, the path length l, and the desired number of output paths k. The algorithm performs iterative DFS with a last-in-first-out *(LIFO)* queue Q to facilitate the search. A minimum priority queue $paths$ is used to store the k least

2. The algorithm explores all possible paths from s to t in G. If the total number of such paths is lower than k, then the algorithm returns all the paths, in the order ranked by their promiscuity values. The algorithm returns an empty output in case there are no paths from s to t in G.

Algorithm 1: Naïve algorithm for finding $\Phi_G(s,t,k)$ and the (up to) l least promiscuous paths $p_{(s,t,k)}$ in G

Data: Knowledge graph G, source node s and target node t in G, path length k, number k of paths from s to t to return.

Result: The l-promiscuity score $\Phi_G(s,t,l)$ and a sorted list of the k least promiscuous paths of length l from s to t.

```
1 begin
2     Q ← InitializeQueue(Q, G, s); // LIFO queue
3     Φ_G(s,t,k) ← +∞ ; paths ← empty queue of length k; // min priority
          queue keyed on p.score
4     while Q ≠ ∅ do
5         m ← Q.pop();
6         Q, paths ← ExtendPath(G, s, t, m, k, Q, paths);
7     if paths not empty then
8         Φ_G(s,t,k) ← paths[1].score;
9     return Φ_G(s,t,k), paths;
```

promiscuous paths[3] seen at any time during the search. The priority queue is keyed on $p.score$ to enable efficient insertion when new paths are found.

The LIFO queue Q is created in line 2; it is initialized with all the neighbors of the node s, as these are the first nodes on any path from s to t in G. For each node n in Q, $n.depth$, $n.score$, and $n.parent$ are set appropriately to 1, 0, and s, respectively. The priority queue $paths$ is created in line 3 as an empty queue of length k.

The main loop for searching G is shown in lines 4–6. In each iteration, the node m from the front of the queue Q is popped (line 5), and its path is extended appropriately (line 6) by calling the procedure $ExtendPath$ sketched in Algorithm 2. (We will provide details on $ExtendPath$ shortly.) Once the queue Q is empty, there are no more paths to consider, so the k least promiscuous paths have been finalized and are stored in the priority queue $paths$. The value of $\Phi_G(s,t,l)$ is extracted in line 8 as the promiscuity score of the first path in the priority queue $paths$, i.e., the score of the least promiscuous path. The promiscuity score $\Phi_G(s,t,l)$ and the k least promiscuous paths are returned in line 9 as the result of Algorithm 1.

3. See footnote 2.

Algorithm 2: Procedure ExtendPath

Data: KG G; source and target nodes s t; current node m in G; path length l; queue Q; least promiscuous paths $paths$.

Result: The queue Q and the extended least promiscuous paths $paths$.

```
1  begin
2      pathScore ← max{degree(m), m.score};
3      N ← neighbors of m in G;
4      if m.depth = k then
5          if t ∈ N then
               // we have found a path of length l from s to t
6              p = extractPath(s, t, m); p.score = pathScore;
7              paths.insert(p); // sorted-order insertion
8      else
9          for n ∈ N do
10             n.depth ← m.depth + 1;
11             n.score ← pathScore;
12             n.parent ← m;
13             Q.push(n);
14     return Q, paths;
```

The procedure $ExtendPath$, shown in Algorithm 2, takes as input the graph G, the source node s, the target node t, the current node under consideration m, the path length k, the LIFO queue Q, and the priority queue $paths$. The current node m was found by exploring some path p of the form $(s, v_1, v_2, \ldots, v_k, m)$. Ideally, p will eventually extend to the target node t, so we compute the score of this path so far $pathScore$, see line 2, as the maximum between the degree of node m and $m.score$, which is the maximum node degree of $v_1, v_2, \ldots, v_k$.

Lines 4–5 check to see if we have found a complete path of length k to the target node t. If so, we *extract* the path p from s to t by repeatedly following the $parent$ attribute of each node, starting with node m, until the start node s is reached (line 6). The promiscuity score of this path $p.score$ is set to the previously computed $pathScore$. We insert the newly found path p into the priority queue $paths$ (line 7). Note that this insertion will place p in its appropriate spot in the priority queue sorted by the path scores, unless $p.score$ is greater than the scores of the k paths currently in $paths$. In the latter case, p is not inserted into the queue, since p is not among the l least promiscuous paths explored so far. If, however, we have not yet reached depth l (line 8), then each

neighbor n of m is pushed onto Q with the appropriate values for $n.depth$, $n.score$, and $n.parent$, see lines 10–13. Either Q or $paths$ may have been updated by Algorithm 2, so both are returned in line 14.

Putting Algorithms 1–2 together appropriately completes the DFS approach and produces as a final result the promiscuity value $\Phi_G(s,t,k)$. The runtime of the naïve algorithm is that of depth first search, i.e., $O(b^{l+1})$, where l is the path length and b is the branching factor of the KG G, i.e., the maximum degree of any node in G.

2.3.2 The Improved-Efficiency Algorithm

Algorithm 3: Fast algorithm for finding the promiscuity score $\Phi_G(s,t,k)$

Data: Knowledge graph G, source node s and target node t in G, path length k, number l of paths from s to t to return.

Result: The k-promiscuity score $\Phi_G(s,t,k)$ and a sorted list of the l least promiscuous paths of length k from s to t.

```
1  begin
2      Q ← InitializeQueue(Q, G, s); // LIFO queue
3      paths ← empty queue of length k; // priority queue keyed on p.score
4      Φ_G(s,t,k) ← +∞; paths[l].score ← +∞;
5      while Q ≠ ∅ do
6          m ← Q.pop();
7          if degree(m) < paths[k].score then
8              Q, paths ← ExtendPath(G, s, t, m, k, Q, paths);
9      if paths not empty then
10         Φ_G(s,t,k) ← paths[1].score;
11     return Φ_G(s,t,k), paths;
```

We now introduce an improved-efficiency algorithm for finding paths $p(s,t,k)$ and the promiscuity score $\Phi_G(s,t,k)$. The algorithm leverages the fact that to find promiscuity scores, we may not need to explore nodes of high degree. By maintaining the lowest promiscuity score of all paths seen so far, we can prune paths whose scores are guaranteed to be higher. Therefore, we eliminate the need to completely explore all paths between s and t. By simply exploring nodes in order of ascending node degree, as opposed to depth-first order, we are guaranteed that the first path

of length k that we find from source node s to target node t will be the path of lowest promiscuity. Therefore, we eliminate the need to continue exploring all other paths between s and t. These claims will be explored in greater detail in Section A.1.3.

Algorithm 3 shows the pseudocode for this approach, which is very similar to that of Algorithm 1 introduced in Section 2.3.1. The key difference occurs within the main loop of lines 5–8. Once we pop the node m from the front of the queue Q, we compare its degree to $paths[l].score$, i.e., to the score of the lth least promiscuous path explored so far. (The value of $paths[k].score$ is initialized to $+\infty$ to account for all the cases in which $paths$ has fewer than l paths.) If the degree of m is greater than or equal to $paths[l].score$, then there is no need to continue exploring this path further, as it is guaranteed by m's existence to have a promiscuity value that is at least as high as the lth least promiscuous path. Thus, we only call the procedure $ExtendPath$ of Algorithm 2 on m if the degree of m is less than $paths[l].score$, see lines 7–8. Effectively, this prunes all the paths that include the node m whose degree exceeds the l-highest known promiscuity score.

The runtime of the improved-efficiency algorithm is in $O(b * p^{l-1} * lg(b * p^{l-1}))$. This runtime shares its worst case with DFS; indeed, if no path can be found between s and t, an exponential number of potential paths must be explored. Recall that the runtime of DFS can be described as linear with respect to the number of nodes and edges, or as exponential with respect to the branching factor. We choose to describe it with respect to the branching factor, because this allows us to incorporate the path length l and the promiscuity score p of the first path from s to t that is found in Algorithm 3. We note in particular that the runtime is inherently dependent on the promiscuity score of the first path found, as this determines the amount of pruning that is done throughout the rest of the search. Observe that p is likely to be significantly less than b, the branching factor of G, in all cases but the worst case. Thus, the improved-efficiency algorithm is likely to provide a substantial speedup over the naïve algorithm. We explore this claim in our experiments, see Section 2.5.5.

The improved algorithm leverages the fact that we only need to analyze nodes with low degree to find minimum promiscuity. If we sort nodes by degree before popping them off the queue

for analysis, we can guarantee that we have found the path with the smallest promiscuity score much earlier than the naïve algorithm. In fact, since the number of neighbors of a node is directly tied to its degree, the promiscuity score serves as an upper bound on the number of paths we must examine. The runtime of our improved algorithm is $O(b * p^{k-1} * lg(b * p^{k-1}))$.

2.3.3 Breadth-First Search

Not surprisingly, both the naïve algorithm and the improved-efficiency algorithm have breadth-first search *(BFS)* (Moore, 1959) equivalents. We note first that the DFS and BFS versions of the naïve algorithm have the same runtime. However, the runtime of the BFS version of the fast algorithm is $O(b * \Phi_G(s, t, k)^k)$; in particular, this is better than the runtime of the DFS version discussed in Section 2.3.2. Unfortunately, the BFS version of the improved-efficiency algorithm has exponential memory requirements in the worst case. Thus, in our work we opt for the DFS version, which has linear memory requirements while still offering a speedup over the naïve algorithm.

2.3.4 Implementation for the Case Study and Experiments

In this section we outline an implementation of the proposed computational approaches; we used this implementation both in the case study presented in Section 2.4 and in the experiments discussed in Section 2.5. For the implementation we chose the format of Neo4j[4] plugins due to their usability, efficiency, and portability advantnages. Neo4j plugins are written in Java and compiled as Jars, and are called in the Neo4j graph database-management system *(DBMS)* in ways that are similar to user-defined functions in SQL. Within the plugin code, our implementations of both the naïve algorithm of Section 2.3.1 and of the improved-efficiency algorithm described in Section 2.3.2 can conveniently access through the provided Neo4j APIs the graph data sets that we used in the case study and in the experiments reported in Sections 2.4–2.5.

4. https://neo4j.com

The plugin that we have developed to implement the proposed algorithms is publicly available; it can be installed via downloading the packaged Jar off of our Github[5]. After the Jar is placed into the *plugins* directory of the Neo4j graph DBMS, the algorithms can be run from the Neo4j Cypher browser using the `CALL` command. We discuss this implementation in more detail in Section A.3.

2.4 Case study

In this section we study how the promiscuity value of a path correlates with its biomedical value. To this end, we present some examples of paths in the ROBOKOP KG (Morton et al., 2019) that connect expert-selected drug-disease pairs. For each such drug-disease pair, the drug is known to be commonly used to treat the disease; we consider whether the intermediate nodes along each path are associated with the drug's treatment of the disease. Pathways reflecting the drug's treatment mechanism are more meaningful to biomedical experts, since they can be used to understand the connection between the drug and disease, see Section 2.1. Accordingly, we rate the extracted pathways by their promiscuity values, and compare the promiscuity value of each path to its biomedical value.

We hypothesize that paths with low promiscuity values would be more meaningful to biomedical experts than other paths, see research task 1 in Section 2.2.3. Following the reasoning presented in Section 2.1 and illustrated via Figure 2.3, we expect lower-promiscuity paths to contain intermediate nodes falling into the same *biological context* as the drug and disease endpoints, see Section 2.1. On the other hand, higher-promiscuity paths would include more divergent intermediates nodes that are not as closely related to the drug and disease endpoints. Intermediate nodes in the same biological context are more relevant to the drug-disease pairs, and are therefore expected to reflect the drug's treatment mechanism in a meaningful way. In this case study, we found evidence corroborating the hypothesis that low-promiscuity paths tend to be more meaningful to biomedical experts.

5. https://github.com/DnlRKorn/promiscuity-procedure

2.4.1 Case-Study Setup

In this case study we used our implementation of Algorithm 3, see Section 2.3.4, to generate in the ROBOKOP KG[6] both low- and high-promiscuity pathways connecting ten well-studied drug-disease pairs that each have a well-known treatment relationship. (The version of the ROBOKOP KG (Morton et al., 2019) that we used contains over 610,000 nodes of 35 types and over 8M relationships of 145 types.) As inputs to the algorithm, for each drug-disease pair we used the ROBOKOP KG as the input graph G, the drug as the source node s, and the corresponding disease as the target node t, with path lengths $l \in \{2, 3, 4\}$ and no bound k on the number of paths, as we wanted to extract paths of both low and high promiscuity. We present the results for paths of length 3 only; the results for paths of lengths 2 and 4 are similar to those shown.

We sorted the paths returned by Algorithm 3 according to their promiscuity values, and presented the top-5 *least promiscuous paths* (i.e., the paths with the five lowest promiscuity values) and top-5 *most promiscuous paths* to the biomedical expert on our team. The expert evaluated each path qualitatively from the biomedical perspective, labeling each path as either *meaningful* or *noisy*. In the context of the hypothesis presented in the beginning of this section (see also research task 1 and the associated hypothesis 1 in Section 2.2.3), we asked the expert to classify paths as *meaningful* when the information provided by the intermediate connections in the path is relevant to the drug's treatment of the target disease and provides meaningful insights into the drug-disease connection. Otherwise, we asked the expert to classify the path as *noisy*.

In alignment with the hypothesis put forward for this study, our expectation was that the biomedical expert would find the low-promiscuity paths to be *meaningful* and paths of high promiscuity to be *noisy*. To this end, we evaluate the expert's analysis using the *correlation score* for each drug-disease pair. We define the correlation score between a drug and disease as the proportion of paths that were classified by the expert according to our expectation, i.e., the total number of both *meaningful low-promiscuity* paths and *noisy high-promiscuity* paths out of the ten paths returned by Algorithm 3 for each drug-disease pair.

6. http://robokopkg.renci.org/browser/

Table 2.1: The correlation scores of 10 drug-disease pairs, i.e., the proportion of of the five least promiscuous paths and five most promiscuous paths that were classified by our biomedical expert as either *meaningful* low-promiscuity paths or *noisy* high-promiscuity paths.

Drug (s)	Disease (t)	*Correlation*
Atorvastatin	Cardiovascular Disease	1.0
Pregabalin	Epilepsy	0.9
Dimethyl Fumarate	Multiple Sclerosis	1.0
Enzalutamide	Prostate Cancer	1.0
Esomeprazole	Gastroesophageal Reflux Disease	1.0
Ibrutinib	B-Cell Chronic Lymphocytic Leukemia	1.0
Pemetrexed	Mesothelioma	0.5
Pomalidomide	Plasma Cell Myeloma	0.8
Rivaroxaban	Pulmonary Embolism	0.7
Sitagliptin	Type 2 Diabetes Mellitus	0.9

The ten selected drug-disease pairs, along with their correlation scores, are presented in Table 2.1. We first notice that half of the drug-disease pairs have the correlation score of 1.0, and eight out of the ten have the correlation score of at least 0.8; that is, our hypothesis is true in most of these cases. At the same time, there are some cases for which the hypothesis does not hold. The biomedical expert indicated that this mainly occurred when the intermediate nodes on the paths were generic highly connected nodes, e.g., nodes that describe the drug's molecular structure, certain genes involved each in many processes, and highly repurposed drugs that are still relevant to the drug-disease relationship.

In the remainder of this section we present the expert's detailed analysis for three drug-disease pairs: (1) *atorvastatin* and *cardiovascular disease* (Section 2.4.2); (2) *pregabalin* and *epilepsy* (Section 2.4.3); and (3) *pemetrexed* and *mesothelioma* (Section 2.4.4). In the exposition, we explore some of the paths used in the case study for these drug-disease pairs and explain the reasoning used by the biomedical expert to classify them. Recall that our goal is to study whether low-promiscuity paths are more likely to include well-supported academic evidence for their intermediate nodes/edges (i. e., are more likely to be biomedically *meaningful)* than high-promiscuity paths. We present our reasoning for why the results for these three pairs align with the

Table 2.2: A sampling of the length-3 paths from atorvastatin to cardiovascular disease; *score* stands for the promiscuity value of the pathway.

s	v_1	v_2	t	Score
Atorvastatin	idiopathic spontaneous coronary artery dissection	non-inflammatory vasculopathy	Cardiovascular Disease	30
Atorvastatin	idiopathic spontaneous coronary artery dissection	aortic aneurysm, familial thoracic 8	Cardiovascular Disease	30
Atorvastatin	anti-neutrophil cytoplasmic antibody-associated vasculitis	predominantly small-vessel vasculitis	Cardiovascular Disease	40
Atorvastatin	anti-neutrophil cytoplasmic antibody-associated vasculitis	microscopic polyangiitis	Cardiovascular Disease	40
Atorvastatin	rare disorder with ptosis	fluticasone	Cardiovascular Disease	366

hypothesis that low-promiscuity paths are likely to be *meaningful* and high-promiscuity paths are likely to be *noisy*.

2.4.2 Exploration of Atorvastatin and Cardiovascular Disease

This section discusses characteristics of paths between the drug *atorvastatin* and the disease *cardiovascular disease* that were analyzed by the expert. (Some of these paths are presented in Table 2.2.) Atorvastatin, brand name Lipitor, belongs to a class of drugs called statins. Statins inhibit the HMG-CoA reductase enzyme, thus slowing the rate-limiting step in the body's pathway to produce cholesterol and other lipids (Lea and McTavish, 1997). Atorvastatin effectively lowers lipid levels in patients with high cholesterol, and is often prescribed to improve the condition of patients with cardiovascular diseases and patients with dysregulated lipid metabolism (hyperlipidemia) (Nelson, 2013).

The intermediate nodes v_1 and v_2 of the first four paths presented in Table 2.2 all belong to the same category of biological entities as "cardovascular disease" and "atorvastatin," as these entities are all related to the heart and the cardiovascular system. Thus, these entities are relevant

Table 2.3: Some of the length-3 paths from pregabalin to epilepsy; *score* stands for promiscuity value.

s	v_1	v_2	t	Score
Pregabalin	body dysmorphic disorder	somatoform disorder	Epilepsy	32
Pregabalin	Spinocerebellar atrophy	carbidopa, levodopa drug combination	Epilepsy	44
Pregabalin	congenital aortic valve stenosis	endocardial fibroelastosis	Epilepsy	56
Pregabalin	gamma-amino acid	gamma-amino-beta-hydroxybutyric acid	Epilepsy	68
Pregabalin	uremia	Alpha 2-HS Glycoprotein AHSG	Epilepsy	431

to the relationship between *atovastatin* and *cardiovascular disease*, and their corresponding paths were classified by the expert as *meaningful*. These first four paths have low promiscuity values between 30 (the promiscuity score for the pair) and 40.

On the other hand, consider the intermediate nodes of the final path shown in Table 2.2. Intermediate node v_1, "rare disorder with ptosis," describes conditions involving facial-nerve damage that can result in ptosis, i.e., drooping of the eyelid (Finsterer, 2003). One such condition is the rare disorder myasthenia gravis (MG) (Conti-Fine et al., 2006), an autoimmune disorder that damages nerves connected to facial muscles. Intermediate node v_2, "fluticasone," is a corticosteroid used to manage nasal allergy symptoms, asthma, and chronic-obstructive pulmonary disease (COPD) (Harding, 1990). These nodes have no obvious meaning relevant to "cardiovascular disease," and therefore do not describe well the relationship between *atorvastatin* and *cardiovascular disease*; as a result, this path was classified as *noisy*. Again, this aligns with our ecxectation, as the promiscuity value of this final path is significantly higher (366) than those of the other four paths presented (30–40).

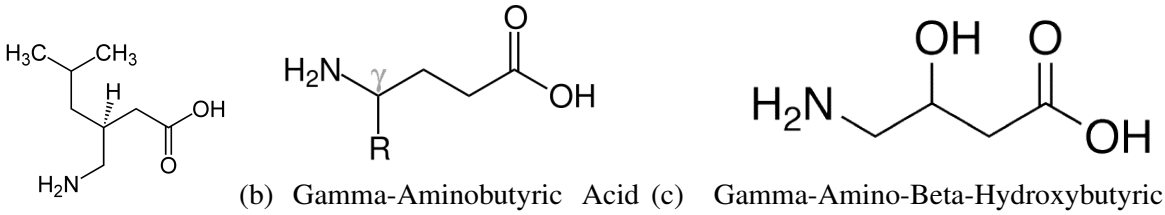
2.4.3 Exploration of Pregabalin and Epilepsy

This section discusses characteristics of paths between the drug *pregabalin* and the disease *epilepsy*; some of the paths are presented in Table 2.3. Pregabalin, brand name Lyrica, is thought to work by binding to presynaptic voltage-gated ion channels in the brain, thereby inhibiting the release of several excitatory neurotransmitters (Ben-Menachem, 2004). This inhibition of neuronal transmission reduces the likelihood of seizures in epileptic patients and dampens neuropathic pain (Ben-Menachem, 2004; Frampton and Foster, 2005).

The first path presented in Table 2.3 connects pregabalin to epilepsy through "body dysmorphic disorder" and "somatoform disorder." A somatoform disorder occurs when a physical symptom or distress presents itself without a clear medical reason (Oyama et al., 2007). The patient truly experiences the symptom, but often the symptom is exacerbated because of the patient's anxiety or obsession with the symptom. Body dysmorphic disorder (Bjornsson et al., 2010) could reasonably fall into this category, and psychogenic non-epileptic seizures are also possible due to somatization, although much more rarely than epileptic seizures (Reuber et al., 2003). Importantly, both intermediate nodes in this path are within the ontological/biological space of neurological conditions, just as "pregabalin" and "epilepsy" are, so this path was classified as *meaningful*. The low promiscuity value (32) of this path aligns with our expectation.

The second path in Table 2.3, which also has low promiscuity value (44), connects pregabalin to epilepsy through "spinocerebellar atrophy" and "carbidopa, levodopa combination." These intermediate nodes refer to a drug combination used to treat Parkinson's disease (Block et al., 1997). Again, as both intermediate nodes remain within the realm of neurological conditions, this path was classified as *meaningful*.

The third path contains intermediate nodes "congenital aortic valve stenosis" and "encardial fibroelastosis," both of which are related to cardiovascular conditions as opposed to neurological ones. Although the promiscuity value of this path (56) is relatively low, the intermediate nodes and the overall pathway do not help us to understand the connection between pregabalin and epilepsy. Thus, this path was classified as *noisy* and does not align with the previously presented



(a) Pregabalin (b) Gamma-Aminobutyric Acid (GABA) (c) Gamma-Amino-Beta-Hydroxybutyric Acid (GHB)

Figure 2.4: Pregabalin (a), Gamma-Amino Acids, like GABA (b), and Gamma-Amino-Hydroxybutyric Acids, like GHB (c), are all closely-related structural analogs with anti-epileptic properties

hypothesis. Here, the intermediate nodes are ontologically/biologically divergent from "pregabalin" and "epilepsy" and are less likely to meaningfully describe the drug-disease relationship. This represents one example of how the expected pattern might fail; such cases were rare in our case study overall.

The fourth path connects "pregabalin" to intermediates "gamma amino acid" and "gamma-amino-hydroxybutyric acid." As show in Figure 2.4, these three molecules are close structural analogs based on the backbone of the inhibitory neurotransmitter gamma-aminobutyric acid (GABA). The two intermediate nodes are also molecules with anti-epileptic activity (Snead III and Gibson, 2005). Thus, these nodes remain very close to "pregabalin" and "epilepsy" in the ontological/biological space, and the path was classified as *meaningful*. The promiscuity value for this pathway is relatively low (68), as expected.

In the final pathway presented in Table 2.3 "pregabalin" is connected to "epilepsy" through "uremia" and "Alpha 2-HS Glycoprotein (AHSG)". Uremia, or uremic pruritus, is a condition of neuropathic pain associated with kidney diseases (Mettang and Kremer, 2015). The protein alpha 2-HS glycoprotein (AHSG) is involved in controlling the calcification of vascular tissue (Jahnen-Dechent et al., 2001). AHSG is reduced in the serum of patients with chronic renal disease (Kishore et al., 1983), and mice deficient in this protein experienced increased renal calcification (Westenfeld et al., 2007), indicating that this gene may be important in uremia and other kidney diseases. However, neither "uremia" nor "Alpha 2-HS Glycoprotein (AHSG)" fall within the ontological/biological space of neurological conditions, and these divergent nodes are unlikely to

Table 2.4: Some of the length-3 paths from pemetrexed to mesothelioma; *score* stands for promiscuity value.

s	v_1	v_2	t	Score
Pemetrexed	malignant pleural mesothelioma	pleural epithelioid mesothelioma	Mesothelioma	55
Pemetrexed	organic heterocyclic compound	Risedronic acid	Mesothelioma	39618
Pemetrexed	organic heterocyclic compound	gefitinib	Mesothelioma	39618
Pemetrexed	organic heterocyclic compound	Temsirolimus	Mesothelioma	39618
Pemetrexed	organic heterocyclic compound	atrazine	Mesothelioma	39618

help describe the pregabalin-epilepsy relationship, so this path was classified as *noisy*. The high promiscuity value (431) of this path aligns with the hypothesis that higher values indicate more divergent realms of knowledge; however, it should be noted that the source of the high value is likely the degree of the "AHSG" gene node. Genes tend to be involved with numerous biological pathways, drug interactions, and diseases, so they often have large node degrees in the ROBOKOP graph; therefore, paths including genes often have relatively high promiscuity values.

2.4.4 Exploration of Pemetrexed and Mesothelioma

This section discusses characteristics of paths between the drug pemetrexed and the disease mesothelioma; some of the paths are presented in Table 2.4. Pemetrexed, brand name Alimta, is a member of the anti-folate drug class (Visentin et al., 2012). Pemetrexed mimics the structure of folate (vitamin B9) and inhibits three folate-dependent enzymes necessary for the biosynthesis of nucleotides (Adjei, 2004). Therefore, pemetrexed interferes with DNA synthesis in rapidly dividing cancer cells and helps treat mesothelioma (Adjei, 2004; Peake, 2009).

The first path between pemetrexed and mesothelioma shown in Table 2.4 contains "malignant pleural mesothelioma" and "pleural epithelioid mesothelioma." Both terms are subclasses of the mesothelioma disease, and therefore fall within the same reasoning space as the the drug and

disease. The corresponding promiscuity value for the path is low (55), demonstrating adherence to the promiscuity-value pattern observed in the other cases.

In the second path, pemetrexed and mesothelioma are linked by "organic heterocyclic compound" and "Risedronic acid". Although the first term, "organic heterocyclic compound", accurately describes pemetrexed as a chemical entity, "organic heterocyclic compound" is a highly promiscuous node that is responsible for the high promiscuity value of the path (39618). The second term, "Risedronic acid", was identified as a potential treatment for mesothelioma by a drug-repositioning study (Dell'Anno et al., 2021) so the term remains relevant to the drug/disease pair. While both "organic heterocyclic compound" and "Risedronic acid" are technically within the pemetrexed-mesothelioma ontological/biological reasoning space, the promiscuity value (39618) is too high, and thus this path does not fit the expected promiscuity-value pattern.

Paths 3-5 also do not fit the expected pattern. Each of these paths contains "organic heterocyclic compound" and a chemical substance (gefitinib, temsirolimus, and atrazine) as intermediate nodes. Gefitinib and temsirolimus have been tested for potential anti-mesothelioma activity (Govindan et al., 2005; Vazakidou et al., 2015), and atrazine has been studied for associations with incidence of cancer (Rusiecki et al., 2004). These three chemical entities fall in the pemetrexed-mesothelioma ontological/biological reasoning space, but all three paths have a high promiscuity value (39618) because of the degree of "organic heterocyclic compound". Thus, these paths do not fit the expected promiscuity-value pattern.

Those paths in Table 2.4 that do not fit the expected promiscuity-value pattern highlight an important consideration about promiscuity values of paths. Nodes of certain types, e.g., "organic heterocyclic compound," are inherently promiscuous because they relate to many other entities across the whole biomedical space. Such entities should be accounted for by experts in their exploration of paths. While the connection of these highly promiscuous nodes may be accurate, the information may not be useful for acquiring biomedical knowledge. This use case highlights a lack of low-promiscuity connections between pemetrexed and mesothelioma in the ROBOKOP KG, as the lowest-ranked paths still contain extremely promiscuous nodes. For the proposed algorithms

to work, the underlying knowledge graph must contain information linking the drug and disease of interest. Thus, this use case in the study shows a circumstance in which merely taking the five lowest-promiscuity paths would not be sufficient. At the same time, the large promiscuity values (39618) of paths 2–5 between pemetrexed and mesothelioma would alert any biomedical expert to treat these paths with suspicion.

2.5 Experimental Results

In this section we present the experimental results that address research tasks 2–4 and the corresponding hypotheses articulated in Section 2.2.3. In summary, the results of our experiments corroborate the three hypotheses and suggest that the proposed algorithmic approaches have value in the computational toolbox of experts in the biomedical domain.

2.5.1 The Data Sets, Data Selection, and Experimental Setup

In the experiments for research tasks 2 and 3 of Section 2.2.3 we used two real-world biomedical knowledge graphs *(KGs),* HetioNet[7] (Section 1.5.2) and ROBOKOP (Section 1.5.1).[8] The version of HetioNet (Himmelstein et al., 2017) that we used contains 47,031 nodes of 11 types and over 2.2M relationships of 24 types. The version of the ROBOKOP KG (Morton et al., 2019) that we used contains over 610,000 nodes of 35 types and over 8M relationships of 145 types. The proposed algorithms have been implemented as described in Section 2.3.4. All the experiments were performed on a server with 32 GBs of memory, 1TB of SSD storage, and an eight-core Intel i7-4770 3.40 GHz CPU. The server ran the Ubuntu 18.04.5 LTS operating system and used the Neo4j[9] graph DBMS version 4.2.1.

Our federated-graph experiments for research task 4 of Section 2.2.3 were performed on Strider, [10] an "autonomous relay agent" within the National Center for Advancing Translational

7. https://neo4j.het.io/browser/
8. http://robokopkg.renci.org/browser/
9. https://neo4j.com
10. https://github.com/ranking-agent/strider

Sciences (NCATS) Biomedical Data Translator program. The distributed KG stored in Strider has approximately 14M nodes and approximately 233M edges, and is accessible via the Strider knowledge portal that enables access to 33+ knowledge providers using a singular entry point and a unified KG data model. In response to a user query, Strider dynamically generates the answer graphs and returns them to the user in the form of JSON documents. The largest constituent KGs available through Strider are COVID-KOP (Korn et al., 2020), RTX-KG2 (Wood et al., 2021), and SPOKE (Nelson et al., 2019).

Recall that studying the utility of the proposed approaches in drug-repurposing applications was a primary target of this project. Accordingly, in setting up the experiments, for each given `(drug,disease)` node pair we selected paths for either the `treats` connection in ROBOKOP or the "Compound treats Disease" `(CtD)` connection in HetioNet. To perform research tasks 2 and 3, we also needed to generate positive and negative drug-disease node pairs. For the positive samples, we randomly sampled the existing "treats"-connected `(drug,disease)` node pairs in the selected KGs. Specifically, let `n` be the drug node, and `m` be the disease node; the triples we selected would be `(n,treats,m)` over the ROBOKOP KG and `(n,CtD,m)` over HetioNet. For the negative (i.e., "not treats") samples, we independently sampled `n` from all possible drug nodes and `m` from all possible disease nodes, and then made sure that we did not accidentally include any positive node pairs, see (Yang et al., 2020) for the details.

2.5.2 Node-Edge Ratio Analysis

We start the analysis by confirming that the selected KGs actually display the promiscuity phenomenon, and are therefore relevant to the experiments and to the case study presented in Section 2.4. Since scale-free networks (Barabási, 2016) are our model for KGs with promiscuity nodes, we checked that the selected KGs exhibit the Pareto principle, a phenomenon present in many scale-free networks. To this end, Figure 2.5 shows the relative cumulative distribution of the node degree in the HetioNet and ROBOKOP KGs. To compute this distribution, we define N_x to be

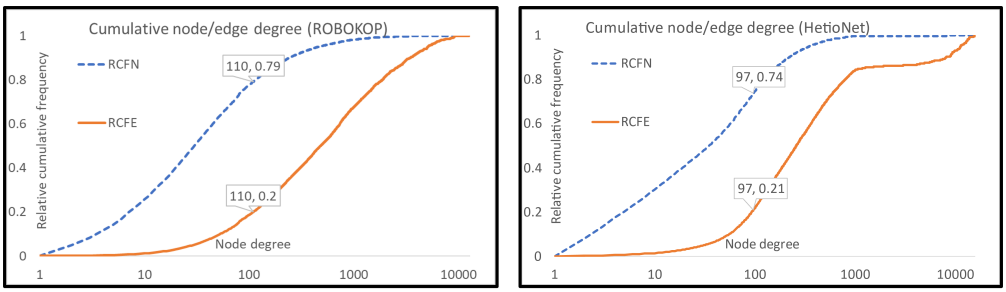


Figure 2.5: Relative cumulative frequency distribution of the node degrees in the ROBOKOP (a) and HetioNet (b) knowledge graphs. The logarithmic x-axes represent the node degrees in each graph, and the y-axes represent the ranges of the relative cumulative frequencies given by Equations (2.4)–(2.5). The upper (dashed blue) line represents the relative cumulative frequency of the nodes whose degree is less than x (RCFN) in each graph, see Equation (2.4); the lower (solid orange) line represents the relative cumulative frequency of the edges with an endpoint whose degree is less than x (RCFE), see Equation (2.5).

the set of all nodes in a KG whose degree is at most x:

$$N_x = \{n \in N \mid degree(n) \leq x\} \,. \tag{2.3}$$

Then, we introduce the definition of *relative cumulative frequency of nodes (RCFN)* in a KG. We define RCFN as the proportion of nodes in the KG whose degree is at most x. (Here, N is the set of all nodes in the KG.)

$$RCFN(x) = \frac{|N_x|}{|N|} \,. \tag{2.4}$$

Finally, we define *relative cumulative frequency of edges (RCFE)* in a KG as half of the total number of edges adjacent to the nodes whose degree is at most x. (Half the total number of edges is used to avoid double-counting edges, i.e., counting an edge once from each endpoint.) Here, E is the set of all edges in the KG.

$$RCFE(x) = \frac{\frac{1}{2}\sum_{n \in N_x} degree(n)}{|E|} \,. \tag{2.5}$$

These formulae allow us to analyze the distribution of edges in a given KG among its nodes. Figure 2.5(a) shows that in the ROBOKOP KG, nodes (blue dotted line) that have fewer than 110 neighbors comprise 80% of all unique nodes in the KG, but only account for 20% of the edges (orange solid

line). This implies that the remaining 20% of the nodes, which have more than 110 neighbors each, account for the remaining 80% of the edges. This phenomenon is often referred to as the 80/20 rule or Pareto principle (Sanders, 1987), in which a small portion of a population contributes a large number to some total. In the case of the HetioNet KG, the proportions are not precisely 80/20, potentially due to the smaller size of the graph, but the Pareto principle is still evident, see Figure 2.5(b). These high-degree nodes are precisely the promiscuous "hubs" that we concern ourselves with in this work. We see this as a confirmation that the KGs chosen for our experiments are, in fact, impacted by the promiscuity phenomenon.

2.5.3 Study of Promiscuity Scores for Positive and Negative Node Pairs for the Treats Connection

Recall (Section 2.2.3) that our hypothesis associated with **research task 2** is that the l-promiscuity scores $\Phi_G(s, t, l)$ for positive node pairs for the *treats* connection in biomedical KGs tend to be significantly lower than the promiscuity scores for negative node pairs for the *treats* connection. To test this hypothesis, we did experiments with the node pairs that had been sampled from the ROBOKOP and HetioNet KGs as described in Section 2.5.1.

To determine whether the l-promiscuity scores $\Phi_G(s, t, l)$ for the sampled positive node pairs are significantly lower than the scores for the negative node pairs, we used two-sample t-tests (Rice, 2006). For each node pair (s, t) we found paths of different lengths l from the `drug` source s to the `disease` target t. Then for each fixed value of l we separated the promiscuity scores obtained for the positive and negative node pairs into two groups, and used a two-sample t-test to check if the mean promiscuity scores from the two groups are significantly different.

We calculated the promiscuity scores as outlined above for 100 positive and 100 negative node pairs for each length l between the values of 2 and 6 (inclusively). To ensure that the existence of the connection between the drug and disease nodes did not bias the scores, we removed the relevant `treats` edges from the Neo4j database before each calculation for the positive node pairs, reintroducing the edges after the completion of the calculation.

The experimental results are shown in Tables 2.5–2.6. We can see that almost all the reported p-values are significantly lower than the standard cutoff value of 0.05[11] for all the values of length l, indicating significant differentiation between the mean promiscuity scores for the positive and negative node pairs in both KGs for each value of l. Observe that all the mean promiscuity scores for the positive node pairs are lower than the corresponding mean negative scores for both HetioNet and ROBOKOP. These results corroborate the claim that the underlying drug-disease connections are at least somewhat reflected by the promiscuity scores.

In Tables 2.5–2.6, we see p-values below 0.05 in most cases; however, in the case of length $l = 2$ on ROBOKOP, we see the p-value 0.053. This high p-value could have been caused by the relatively larger standard deviation for that path length on ROBOKOP. The high variance could stem from the somewhat surprising sparsity of length-2 paths in that KG. Indeed, we found that the drug and disease nodes in many of the positive and negative samples in ROBOKOP are not connected by paths with a single intermediate node in each (i.e., length-two paths). Looking deeper into this situation, we found that 30% of our sampled positive node pairs had no length-two path connections, and neither had 82% of the negative node pairs. When gathering data for our analysis, whenever we sampled an unconnected node pair, we would re-sample until we found a connected pair. This procedure could have led to a biased result for length-two paths.

Despite the slightly larger p-value for length-two path cases in ROBOKOP, the value is still very close to 0.05. (Note that other studies in the literature use 0.1 as their criteria for rejecting the null hypothesis in tests.) In addition, there is a much higher variation in the promiscuity scores for ROBOKOP as compared with HetioNet. This effect is aligned with the fact that ROBOKOP is significantly larger than HetioNet (610K vs 40K nodes).

To summarize, the experiments reported in this section corroborate, via the ROBOKOP and HetioNet KGs, the hypothesis associated with **research task 2** of Section 2.2.3 node pairs for the *treats* connection in biomedical KGs.

11. If the p-value of a two-sample t-test was less than 0.05, we take it to represent that the mean values of the groups are significantly different.

Table 2.5: The means (standard deviations) of the promiscuity scores of the 100 sampled positive and 100 sampled negative drug-disease node pairs for the HetioNet KG, along with the p-values of two-sample t-tests run on the two distributions.

Path length (l)	The positive pairs	The negative pairs	The p-values
2	158 (115)	384 (288)	$<$.00001
3	70 (32)	125 (59)	$<$.00001
4	68 (22)	88 (28)	$<$.00001
5	53 (24)	66 (15)	$<$.00001
6	53 (11)	61 (12)	0.005

Table 2.6: The means (standard deviations) of the promiscuity scores of the 100 sampled positive and 100 sampled negative drug-disease node pairs for the ROBOKOP KG, along with the p-values of two-sample t-tests run on the two distributions.

Path length (l)	The positive pairs	The negative pairs	The p-values
2	516 (822)	3,210 (13,800)	0.053
3	108 (124)	2,260 (5,700)	$<$.00001
4	143 (213)	891 (3,060)	0.017
5	67 (68)	172 (154)	$<$.00001
6	61 (42)	112 (69)	0.005

2.5.4 Node-Pair Prediction

In Section 2.5.3 we saw that l-promiscuity scores $\Phi_G(s, t, l)$ for positive node pairs for the `treats` connection can be significantly lower than l-promiscuity scores for negative node pairs for the same connection. We now study whether the score difference between the positive and negative pairs is strong enough to allow us to use the scores to *find* positive `treats` connections between nodes for drugs and diseases of interest. In other words, in this section we focus on research task 3 formulated in Section 2.2.3, that is, on determining whether a classifier trained by l-promiscuity scores $\Phi_G(s, t, l)$ can correctly predict positive and negative node pairs for some criterion. We consider this task for the `treats` connection between the drugs and diseases of interest.

Similarly to our experimental setup for research task 2 (see Section 2.5.3), for task 3 we used the sampling procedure described in Section 2.5.1 to find the positive and negative node pairs with their corresponding l-promiscuity scores. For this experiment we sampled a total of 500

positive and 500 negative node pairs. The positive (negative, respectively) node pairs were collected into the set $\mathcal{P}$ (the set $\mathcal{N}$, respectively).

In training a classifier, using a single attribute (feature), such as the promiscuity score for a particular path length for a node pair, is unlikely to lead to adequate prediction quality. To address this problem, our strategy in classifier training was to use multiple l-promiscuity scores, one score for each value of path length l within a range of lengths, as the features for each drug-disease node pair. For example, a node pair (s, t) could have $\Phi_G(s, t, 3)$, $\Phi_G(s, t, 4)$, $\Phi_G(s, t, 5)$, and $\Phi_G(s, t, 6)$ as its features. The ground-truth positive class labels (with value 1 in our experimental setup) were assigned to the positive node pairs, and the negative class labels (with value 0) were assigned to the negative node pairs. In this experiment we excluded the cases of path length $l = 2$ due to the sparsity of length-two paths in the ROBOKOP KG, see Section 2.5.3 for a discussion.

For the experiments, we used the promiscuity-score features and the corresponding ground-truth class labels to train a binary random-forest classifier (Ho, 1995). That is, we used the promiscuity scores as the independent variables and the ground-truth labels as the dependent variables in training the classifier. The classifier that we used had been generated using the default parameters of the Python Scikit-Learn machine-learning library (Pedregosa et al., 2011).

To be more specific, given the training set and testing set of drug-disease node pairs, the input data for the classifier training were the promiscuity scores $\Phi_G(s, t, 3)$, $\Phi_G(s, t, 4)$, $\Phi_G(s, t, 5)$, and $\Phi_G(s, t, 6)$ for each pair and the corresponding ground-truth labels, 1 for the positive pairs and 0 otherwise. In the testing process, only the promiscuity scores $\Phi_G(s, t, 3)$, $\Phi_G(s, t, 4)$, $\Phi_G(s, t, 5)$, and $\Phi_G(s, t, 6)$ would be provided for each node pair in the test set, and the classifier would predict whether the pair has the positive or negative class label. In the sequel we will denote the predicted class of node pair p by $class(p)$.

To evaluate the classification quality, we computed the precision and recall as defined in Eq. (2.6)–(2.7). The former metric indicates the proportion of total classifier-predicted positive node pairs that are true positive node pairs, and the latter indicates the proportion of total true positive node pairs that are predicted positive node pairs.

$$precision = \frac{|\{p \in \mathcal{P} : class(p) = positive\}|}{|\{p \in \mathcal{P} \cup \mathcal{N} : class(p) = positive\}|} \,. \tag{2.6}$$

$$recall = \frac{|\{p \in \mathcal{P} : class(p) = positive\}|}{|\mathcal{P}|} \,. \tag{2.7}$$

Further, we computed accuracy as defined in Eq. (2.8), to indicate the proportion of correctly predicted node pairs. We also computed the F1-score, which is the harmonic mean of precision and recall, see Eq. (2.9).

$$accuracy = \frac{|\{p \in \mathcal{P} : class(p) = positive\} \cup \{p \in \mathcal{N} : class(p) = negative\}|}{|\mathcal{P} \cup \mathcal{N}|} \,. \tag{2.8}$$

$$F1\text{-}score = \frac{2 \cdot precision \cdot recall}{precision + recall} \,. \tag{2.9}$$

From the drug-disease node pairs sampled as discussed above, we randomly selected 80% as the testing data set and used the remaining 20% of the pairs as the training data. The results of the random-forest classifier for these data are presented in the confusion-matrix form in Tables 2.7–2.8. Based on these results, for the HetioNet KG we obtained a precision of 0.72, a recall of 0.8, an accuracy of 0.75, and an F1-score of 0.76. For the ROBOKOP KG, the results provided a precision value of 0.87, a recall of 0.81, an accuracy of 0.845, and an F1-score of 0.84.

We now recall **hypothesis 3** formulated in Section 2.2.3: Given l-promiscuity scores for testing-set node pairs in a biomedical KG and the ground-truth (positive or negative) labels for the pairs, a classifier trained on a training set of positive and negative node pairs for the *treats* connection in the KG tends to correctly predict whether the testing-set node pairs are positive or negative. In the above experiment, we saw an overall good performance for the prediction task on the HetioNet and ROBOKOP KGs, with the prediction result for ROBOKOP looking better than the prediction result for HetioNet. This could be the result of the smaller size of the HetioNet KG,

Table 2.7: The confusion matrix for the results of the binary random-forest classifier with the HetioNet KG.

	Predicted positive	Predicted negative
Actual positive	80	20
Actual negative	31	69

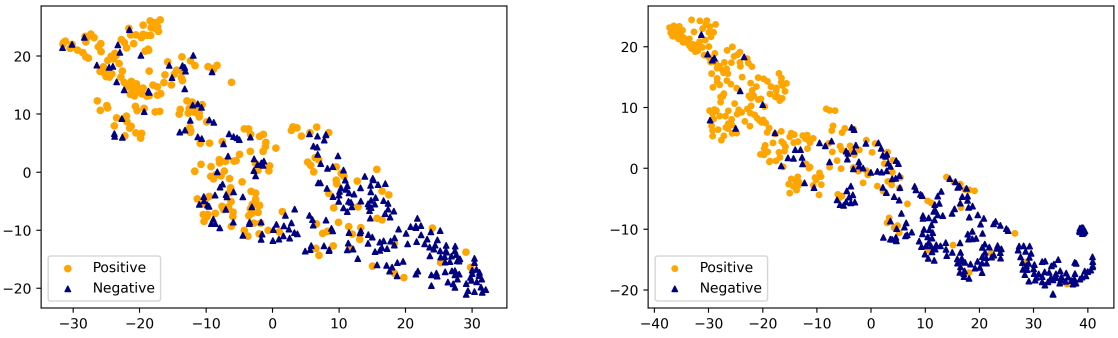
Table 2.8: The confusion matrix for the results of the binary random-forest classifier with the ROBOKOP KG.

	Predicted positive	Predicted negative
Actual positive	81	19
Actual negative	12	88

which has many fewer nodes and edges than ROBOKOP. This would potentially limit the ability of intermediate pathways that may otherwise prove interesting and predictively useful to appear as promiscuous pathways.

Figure 2.6 shows two-dimensional (2D) t-SNE (Van der Maaten and Hinton, 2008) visualization plots of the promiscuity scores of the positive and negative node pairs used in the experiment. (The t-SNE algorithm is a dimensionality-reduction technique for projecting high-dimensional vectors onto low-dimensional spaces, to enable better understanding of the relative distances between the data points.) We used the features obtained for the node pairs (i.e., $\Phi_G(s,t,3)$ through $\Phi_G(s,t,6)$) as the input. In the Figure we can see a projection of the features on a 2D-space, so that we could easily see the distances between those node pairs with regard to the promiscuity scores.

In the projections shown in Figure 2.6 it seems that in both KGs the two groups, positive and negative, cannot be separated clearly using a simple hyperplane. At the same time, trends can be observed in the clustering of the data, indicating weak separation between the positive and negative node pairs. As the hypothesis-generating process for drug repurposing is an inherently difficult task in the biomedical domain, weak separation could still be valuable to biomedical experts. We can see that the points in two groups (positive and negative) of node pairs diverge in the opposite directions, indicating that the proposed concept of promiscuity scores has the potential to be a significant factor in predicting whether a drug-disease node pair is positive or not.



(a) HetioNet (b) ROBOKOP

Figure 2.6: 2D t-SNE projections (Van der Maaten and Hinton, 2008) of the promiscuity scores of the positive (orange circles) and negative (blue triangles) node pairs for HetioNet (a) and ROBOKOP (b). We can observe *weak* separation of these two groups for both KGs. As hypothesis generation for drug repurposing is inherently difficult, weak separation may still be of value to biomedical experts.

2.5.5 Performance of the Proposed Approaches

In this experiment we test the performance of the algorithms presented in Section 2.3. We focus on the execution time for (1) the naïve depth-first (DFS) version of the proposed approach (Section 2.3.1), its (2) improved-efficiency DFS version (Section 2.3.2), and (3) the improved-efficiency breadth-first (BFS) algorithm (Section 2.3.3).

For this experiment we sampled 1000 positive drug-disease node pairs on the ROBOKOP KGs in the way described in Section 2.5.1. Recall that in a positive drug-disease pair, a `treats` edge is present in the KG between the nodes in the pair. We then measured the runtimes taken by the three algorithms to calculate the l-promiscuity scores for paths of length l=3 between the nodes in each sampled pair.

The total runtimes in seconds are shown in Figure 2.7. (For each promiscuity-score value in the range shown, Figure 2.7 reports, on the logarithmic Y-axis, the average of the total runtimes for all the node pairs with that value.) We can see that the runtimes of the naïve version of the algorithm dominate those for both improved-efficiency versions. Among those, the improved-efficiency DFS

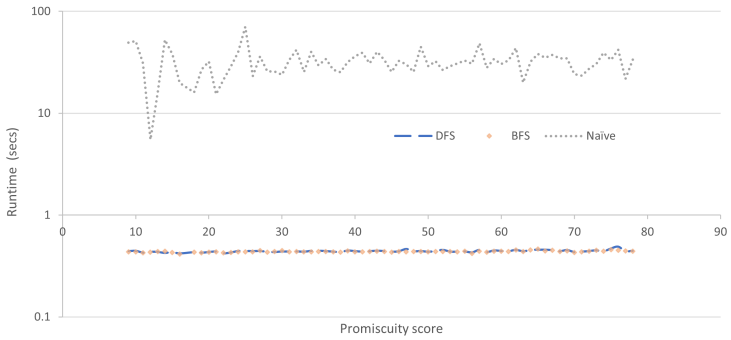


Figure 2.7: Runtimes in seconds for variants of the algorithms for calculating promiscuity scores for `drug-disease` node pairs in the ROBOKOP KG for path length $l = 3$. The x-axis shows the promiscuity scores; the y-axis is logarithmic.

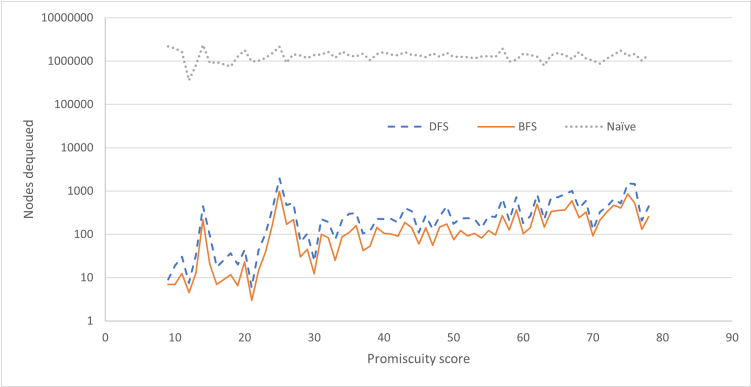


Figure 2.8: Numbers of nodes dequeued for variants of the algorithms for calculating promiscuity scores for `drug-disease` node pairs in the ROBOKOP KG for path length $l = 3$. The x-axis shows the promiscuity scores; the y-axis is logarithmic.

algorithm is marked as "DFS" in the Figure, and the improved-efficiency BFS algorithm is marked as "BFS."

The above results corroborate our claim that the runtime performance of both proposed improved-performance approaches is overall acceptable. At the same time, we found that capturing the raw execution times as reported in Figure 2.7 is not very reliable with our original implementation of the algorithms, as the Neo4j plugin engine introduces significant startup overhead and variance into the execution time, especially for smaller path queries. Thus, any insights that could be gained from the results reported in Figure 2.7 are not necessarily reliable either. To isolate the effects of the Neo4j engine, we modified the existing plugin created as reported in Section 2.3.4, by introducing

the ability to track the number of nodes dequeued during the runs of the algorithms. This number, being an objective measure of the amount of computation that is needed to find the promiscuity score for a node pair in a KG, serves as a practical proxy for the total runtime of the algorithms.

The results for the modified versions of the proposed algorithms are reported in Figure 2.8. (For each l-promiscuity-score value $\Phi_G(s, t, l)$ in the range shown, Figure 2.8 reports, on the logarithmic Y-axis, the average of the total numbers of nodes dequeued in the calculation of $\Phi_G(s, t, l)$ for all the node pairs with that value of $\Phi_G(s, t, l)$.) Perhaps not surprisingly, we can see again that the totals for the naïve DFS version of the algorithm dominate those for both improved-efficiency versions. Indeed, recall that to find the l-promiscuity score for a node pair, the naïve versions of the proposed approaches must examine all the paths of length l between the source and target nodes in the pair, while the improved-efficiency algorithms can selectively prune paths. Observe the upward trend in the numbers of nodes dequeued for both improved-efficiency algorithms. Intuitively, as the value of the promiscuity score increases, the number of paths that must be explored between the source and target nodes for a node pair tends to also increase, raising the number of nodes that need to be dequeued.

In addition to finding that both improved-efficiency algorithms performed substantially better in the experiment than the naïve DFS method, we found that the naïve algorithm would take inordinately long to run for path lengths $l = 5$ and $l = 6$ on ROBOKOP, due to the high numbers of possible paths in the KG for those values of l. (For instance, the algorithm ran for multiple days for one specific drug-disease pair.) Moreover, the naïve approach caused memory-overflow issues in some of those cases. These issues did not arise for the improved-efficiency algorithms, which were showing reasonable performance in those settings, finding the correct promiscuity scores even in the presence of massive numbers of possible paths for some node pairs. As seen in Sections 2.5.3–2.5.4, using those approaches we were able to compute the l-promiscuity scores for a range of values of path length l.

Returning to the results shown in Figure 2.8, we found that, on average, the naïve approach required 25000x more node-dequeue events than the improved-efficiency DFS approach and 46000x

more dequeue events than the improved-efficiency BFS approach. This supports our hypothesis that the proposed improved-efficiency algorithms can be substantially more efficient than their naïve versions at finding promiscuity scores and paths for node pairs in a knowledge-graph environment. Further, the issues described in the preceding paragraph suggest that calculating promiscuity scores for node pairs at high path lengths may be computationally infeasible without the proposed improved-efficiency algorithms.

2.5.6 Efficiency Results for the Strider Federated Knowledge-Graph System

In our final experiment we consider the setting of distributed (as opposed to centralized) knowledge graphs. We work with the Strider system described in Section 2.5.1 to look at the efficiency of forming partial answers to path queries *(partial paths)* in individual KGs, which are then combined into the final path-query answers that are returned to the user. We compare the efficiency of finding partial paths with low promiscuity values using the proposed algorithms with that of the default option of finding any partial paths. Our claim is that using low-promiscuity partial paths in path-query answering in a federated KG system may add to the quality of the final answers returned to the user with acceptable overhead.

Our experimental setting was as follows. As the measure of efficiency of a given query-processing method in the Strider system, we used the number of database (i.e., individual-KG) calls made by Strider to find partial paths during the processing of a given path query. The query-evaluation method that we used as the baseline is the default Strider approach that we call *first come first serve (FCFS)*. In this approach, to find paths between a pair of graph nodes the federated KG is explored via depth-first search, with each path being returned to the user as soon as it has been assembled from partial paths.

The alternative query-processing method that we looked at in the experiment is based on our proposed path-finding approaches presented in Section 2.3. We have implemented the approaches in Python on top of the pre-existing Strider architecture. The key difference of these approaches from the baseline is in changing the order in which Strider would explore the individual

KGs to assemble a path query from partial paths, based on the order of node exploration prescribed by each version of the proposed algorithm.

The BFS and DFS versions of our algorithms provide a tradeoff between memory usage and the number of nodes dequeued and explored by each version. In the federated-KG setting, the number of node dequeues translates into the number of database calls. Our focus in this experiment was on minimizing the number of database calls in the computation of paths between a given pair of nodes. Accordingly, we chose the improved-efficiency BFS version of the algorithm, as it requires fewer node dequeues per path than the alternatives.

For the experiment we sampled 100 positive and 100 negative drug-disease pairs. In contrast with the negative pairs, the positive pairs comprised drugs and diseases that have an explicitly stated *treats* relationship in the Strider federated KG. For each drug-disease pair, we asked Strider to return n answer paths using either our improved-efficiency BFS approach or the baseline approach, with n ranging between 1 and 40. (See Figure 2.9.) The path length for the improved-efficiency BFS approach was set to $l = 3$.

The results of the experiment are presented in Figure 2.9, which shows the difference in the number of database calls required to construct the set of n answer paths for each value of n between 1 and 40. (For each value of n in the range shown, Figure 2.9 reports the average of the total numbers of database calls performed in the calculation of the n answer paths for all the node pairs tested with that value of n.) For instance, for $n = 5$ we see that, on average, it takes the improved-efficiency BFS algorithm 128 database calls to build the five answer paths. This is a 21% increase over the (on average) 106 database calls made by the baseline FCFS approach. Recall that the partial paths discovered by the proposed approaches may be of higher biomedical interest to experts; the overhead for the discovery of such partial paths at $n = 5$ in our experiment is 21%. We envision that in future implementations of federated knowledge graphs, multiple algorithms for construction of partial paths may be presented to the user, enabling a tradeoff in query performance versus quality of the answers. Note that, as the number n of user-requested answer paths increases, the performance advantage of the baseline FCFS approach over the proposed improved-efficiency

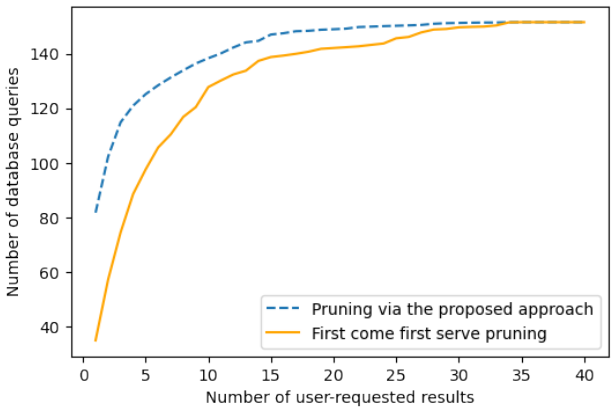


Figure 2.9: Numbers of database calls required to construct in the Strider federated KG the specified number of answer paths for the 200 sampled drug-disease pairs. The results are shown for the proposed improved-efficiency BFS algorithm (blue dashed line) and for the baseline FCFS algorithm (orange solid line).

BFS algorithm tapers off. This is due to the sets of partial paths having higher overlaps between the two approaches, as the ever higher shares of all the possible distinct partial paths are returned from the individual KGs in the Strider federation in the construction of the path answers.

2.6 Limitation

In this chapter, we discuss how the issues of high promiscuity in pathways could negatively impact drug discovery with KGs. Despite the exceptional performance of pathways with low promiscuity in our tasks, some other real-world problems might have more interesting results on pathways with high promiscuity. As we discussed in Section 2.1, in the case of "Dirty drugs (Roth et al., 2004)," their benefit comes from their high interactivity with genes.

For example, if the problem is close to a popular entity, e.g. COVID-19, the promiscuity score would be extremely high as long as the pathway regarding the target entity passes through this node. In such case, a pathway with a low promiscuity score does not provide a more meaningful path as compared to a high scoring pathway. For these examples, the promiscuous nodes are required for the returned pathways to be of interest to the researchers, and thus our methodology would be inappropriate for these cases.

It's possible to think about the example in citation network: for two paths with the same pattern *Author-Paper-Author*, the higher promiscuity path might represent the more citations on the collaborative paper, while the lower promiscuity path might represent the lesser citations on the collaborative paper between the same two authors. In this case, the lower-score path does not represent more meaningful path. However, in the same case, if the papers also connect more author nodes. The promiscuity will be more meaningful for the same path because it represents a paper was collaborated by fewer authors and we found this target Author. In this case, the lower-score path do represent more meaningful path.

2.7 Conclusion

In the context of finding mechanistically justified drug-disease paths in knowledge graphs (KGs), we focused in this thesis on the challenge of processing of promiscuous KG nodes, that is, nodes that are associated with numerous relationships that may not be unique or indicative of the node properties. Specifically, the presence of promiscuous nodes on drug-disease graph paths can dilute the semanticity of the paths. To address the promiscuous-node challenge, we introduced the notion of promiscuity values and scores for nodes and paths in KGs, and presented a suite of algorithms that return to the user the specified number of paths with the lowest promiscuity values between the given pair of nodes. We reported the results of a case study that indicate that paths with low promiscuity values could be meaningful and of interest for biomedical experts in drug repurposing. We also presented experimental results that suggest that the proposed algorithms can be efficient even in distributed KGs, as well as effective at discriminating between related and unrelated drug-disease pairs and at predicting drug-repurposing relationships between drugs and diseases. As the proposed algorithms are domain- and task-independent, we are currently looking into the potential of their application to other tasks and domains, as well as into modifying the approaches to make them effectively applicable to other scenarios.

2.8 Future Work

One aspect of the promiscuity algorithm we discussed in the introduction was the utility of our algorithm working on generalized biomedical knowledge graphs without modification, such as ROBOKOP. The construction of dynamic specialized knowledge graphs which remove all nodes commonly thought to be promiscuous may produce interesting results. This method would either need to fork an existing graph database and create an entirely new database with only nodes that may produce interesting pathways. Another approach may be a dynamic filtering step, which at runtime of the process, takes in the source biomedical knowledge graph and generates a subset of the graph; with some pruning algorithm removing undesired nodes.

The embedding methods discussed in the related work section rely on neighborhoods of nodes and random walks to determine which how nodes should relate to each other. This leads to nodes with more neighbors dominating embedding of graphs. The dominance of promiscuous nodes can be viewed as a version of the "friendship paradox", which is a phenomenon in social networks which cause the average person to have fewer friends then their friends.(Field, 1991) Similarly, a node will likely have a much smaller degree than that of its neighbors. These nodes end up dominating the embedding, forcing all nodes to cluster near nodes with the highest degree. In many cases, this is a desired phenomenon of existing embedding algorithms, as large central nodes with many edges should be important to the meaning of the graph.

We theorize that the problem with current embedding methods relates to their methods of sampling the graph. Node2Vec and its derivative methods use a random walk.(Grover and Leskovec, 2016) This node2vec random walk provides the user with parameters which enable it to be biased and interpolated in either a "Depth-First Search" and a "Breath-First Search" manor.

An interesting approach towards tackling this friendship paradox issue would be introducing a graph embedding methodology which can account for these promiscuous nodes. In this methodology, the embedding random walks could be restricted to either heavily down-weight promiscuous nodes and ignore them entirely. Once such an algorithm exists, it would be exciting

to explore how the embeddings produced differ from traditional embeddings. Exploration of rare diseases pathways, with complex understudied mechanisms are a prime area which such methods could provide utility.

2.8.1 Embedding Methods

In existing random walk based graph embedding algorithms, a collection of random walks are utilized as the primary method of constructing, updating, and representing the underlying graph. Through this representation, nodes of small degree are underrepresented due to natural biasing which occurs during random walks, in which the neighborhoods of nodes of low degree become absorbed by promiscuous nodes. We aim to correct for this to construct embeddings which more fully capture information about low degree nodes.

In seeking to address the issue of biased random walks around promiscuous nodes we seek to make the following modifications to existing graph embedding algorithms. The primary objective is to increase the information surround smaller nodes with fewer degrees and prevent the leakage away of random walks seen in Figure 2.10. In the existing node2vec embedding algorithms there exists a sampling step for the generation of random walks. This sampling would select with knowledge of promiscuous nodes to more fully represent knowledge graph nodes.

We seek to control the effect of promiscuous nodes from embedding methodologies. To enable higher order clustering for hypothesis generation, we propose the creation of a novel embedding methodology PAEV (Promiscuity Aware Embedding Vectors). This embedding method will be designed around the goal of promoting connections with small local clusters of nodes, and to algorithmically ignoring promiscuous nodes. Our algorithm will leverage a variety of weighting functions to down-weight the effect of promiscuous nodes, including quadratic, gaussian, and cut-off filters to control the effect of promiscuous nodes on embeddings of its neighbors.

In our exploration of promiscuity filtered pathways, we found novel connections. By utilizing these insights, this embedding algorithm will hopefully provide a unique way to view the complexity of modern knowledge graphs. Once the PAEVs have been created for nodes of a

graph; they can be utilized by clustering methods, such as SLINK(Sibson, 1973) and k-Means(Likas et al., 2003). Finding these clusters should indicate nodes existing in similar areas of a graph, and hopefully these groups have interesting mechanistical similarity which can be further analyzed and utilized by domain experts. An ideal case would be these clusters enabling the creation of high-level categories for ontological objects in a field. Giving further understanding to mechanistic behaviors in these domains.

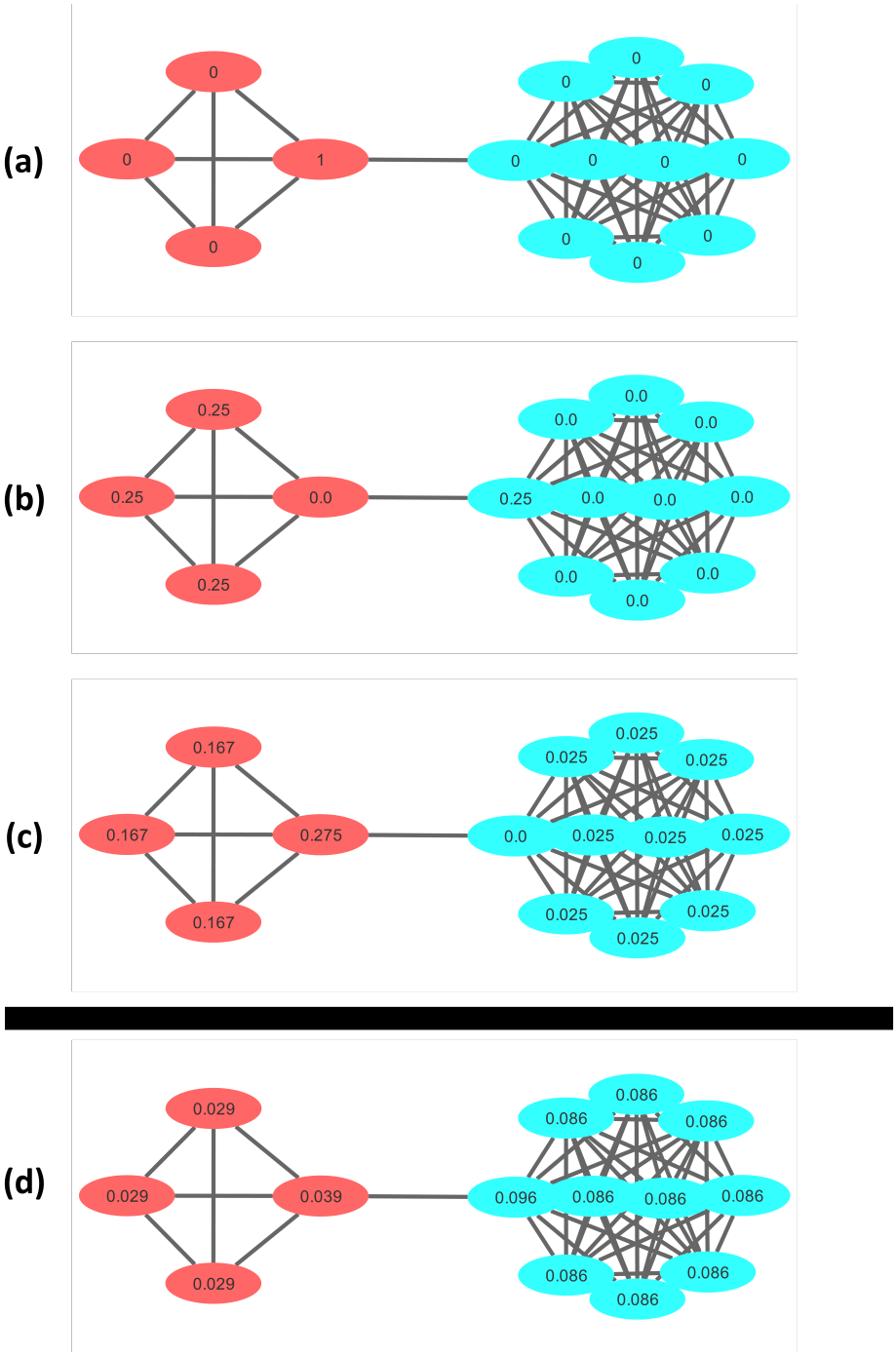


Figure 2.10: In this figure we show a simple example of a random walk between two cliques of nodes. On the left we see a 4-clique and on the right a 10 clique. We have one node from each clique with an edge between them. We simulate the probabilities of traversing any node in a uniform random walk. In (a) we take the *first* step in selecting the node in the 4-clique with an edge to the 10-clique. In (b), we see the probabilities on the *second* step of the random walk. (c) shows the probabilities of the *third* step. (d) shows this process run nearly to convergence, the *eightieth* step. We see that as this random walk runs further along, the probabilities of walking nodes are leeched away from the 4-clique until they are almost entirely gathered in the 10-clique.

CHAPTER 3

Semantic Graph Pathways

3.1 Introduction

Exploration of biomedical knowledge graphs has become a challenging and evolving problem. These graphs have been developed and deployed at large scale use, examples can be seen such as ROBOKOP (Section 1.5.1) and HetioNet (Section 1.5.2). For the first time in the history of biomedical science, we have gathered a large collection of information on drug/disease interactions into a singular interoperable database. This provides us a chance to try expanding our definitions of how drugs and diseases interact. By using these databases a knowledge source to draw from, we can seek to describe these interactions through complex pathways which traverse a knowledge graph.

Traditionally the effect drugs have on the human body have been classified in two ways. The first is by what *biological target* it interacts with and the second is by the *Mechanism of Action* (MOA). Wermuth defines biological targets as "molecular structures, chemically definable by at least a molecular mass, that will undergo a specific interaction with chemicals that we call drugs because they are administered to treat or diagnose a disease."(Van den Broeck, 2015) The target for many drugs is well defined, as the process can be validated through an assay of the drug against the target. Targets are useful and convenient as they provide a high-level classification of chemicals which concerns how they affect the human body.

This way of representing queries is very useful in enabling our ability to utilize knowledge graphs for more complex forms of hypothesis generation. Domain experts can easily create query patterns from their own specialized knowledge in a field. creating patterns that may have particularly important meaning when answered by a well-maintained knowledge graph.

Chapter 2 explored the problem of finding meaningful pathways in a knowledge graph utilizing the graph environment with little context for node and relationship labels. The promiscuity algorithm proposed within that chapter helped produce pathways of biomedical relevance. This chapter addresses the *semantic query* problem. We seek to explore the path search problem from a different angle. As utilization of biomedical knowledge graphs increases, we see that existing paradigms for exploring the pathways of these graphs in a *semantically meaningful* fashion is insufficient. Existing strategies for pathway exploration in knowledge graphs rely upon either patterns found through machine learning or those created by domain-experts working on the graphs. We claim that integrating the efforts of these domain-experts will help to fully capture the underlying semantic meaning contained in these graphs.

Here, we seek to to go in depth in capturing the semantic meaning in one important biomedical case, that of drug $\rightarrow$ disease mechanisms. In Section 3.2, we describe the *Clinical Outcome Pathway*, a complex metapath constructed to help elucidate the relationships between drugs and diseases explore biomedical knowledge graphs.

In Section 3.3 we treat Clinical Outcome Pathways as a case study in building complex metapaths. We then explore the broader implications of constructing metapaths on top of a knowledge graphs, and how we can generalize pathways in knowledge graphs to capture semantic meaning. We formalize the concept as the idea of *semantically query patterns*.

3.2 Clinical Outcome Pathways

Semantically meaningful graph patterns have been defined in the abstract case as a "domain-specific knowledge captured in the format of a semantic graph, which, when analyzed in the context of the domain, has semantic meaning." But to define a flexible pattern that can be leveraged to get domain knowledge is a substantial challenge. The challenge comes from several factors

1. **Domain-specific knowledge** contains a great deal of complexity and edge cases. Capturing this information into a single entity, such as a graph pattern, is a difficult task,

2. Domain-specific knowledge graphs have only been in existence for a few years and have only recently been of high enough quality for such an exercise to be fruitful, and

3. Integration between the computational community who create these tools and Subject Matter Experts (SMEs) who could benefit from these tools has been lacking.

By working with domain experts in the biomedical field, we have created a semantic graph pattern of interest for the bioinformatics community. Our goal is to create this pattern that other domain experts may use as a guide, applying a similar methodology to their own fields to create novel semantically relevant graph patterns that may be integrated with future computer science research.

3.2.1 Related Work

The term Clinical Outcome Pathway has been used in the literature on occasion. For instance, in 2017, the National Center for Advancing Translational Sciences (NCATS) used the term COPs in a 'Request For Applications' announcement.(for Advancing Translational Sciences, 2017) In 2018, we used this term in an application note on Chemotext,(Capuzzi et al., 2018) an online tool for uncovering relationships between ontological terms in the scientific literature, as annotated in PubMed. In addition, a simplified instance of a COPs was implemented in ROBOKOP (Section 1.5.1) to support the elucidation of biological pathways of relevance to drug effects. This implementation of the COP took the form of a query through through the knowledge graph of the form:

$$\begin{aligned}(\textbf{Chemical}) \rightarrow (\textbf{Gene}) \rightarrow &(\textbf{Biological Process}) \rightarrow (\textbf{Cell}) \rightarrow (\textbf{Anatomical Entity}) \rightarrow \\ &\rightarrow (\textbf{Phenotypic Feature}) \rightarrow (\textbf{Disease})\end{aligned} \tag{3.1}$$

We show an example ROBOKOP COP query in Figure 3.1. To explore how to generate this query, please go to the link: `https://robokop.renci.org/simple/question/`.

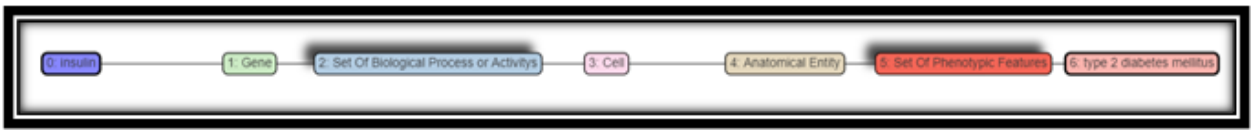


Figure 3.1: An example Clinical Outcome Pathway (COP). This COP aims to explore the connection between insulin and type two diabetes. This COP reflects the version currently implemented into the ROBOKOP graph database.

Within that webpage, follow the option *Template*, and select the template "COP for $name1$ ($identifier1$) and $name2$ ($identifier2$)?"

3.2.2 Introducing Clinical Outcome Pathways

Historically, the groupings of drugs by chemical class, primary target(s), and/or mechanism of action (MOA), have served as a convenient means of classifying drug effects and corresponding indications.(Avram et al., 2021; Kinney et al., 2019; Mestres et al., 2008) Here, we propose a more informative description of a chain of chemical-biological interactions underlying clinical effects of drugs, which we formally describe as **Clinical Outcome Pathways (COP)**.

The term COPs has been used occasionally in recent literature. For instance, in 2017, the National Center for Advancing Translational Sciences (NCATS) used the term in a 'Request For Applications' announcement.(for Advancing Translational Sciences, 2017) In 2018, we used this term in an application note on Chemotext, (Capuzzi et al., 2018) an online tool for uncovering relationships between ontological terms in the scientific literature, as annotated in PubMed. However, to the best of our knowledge, this paper presents the first attempt to define COP as an overarching biomedical concept. Furthermore, we implemented a specific type of COP-based queries in ROBOKOP (Reasoning Over Biomedical Objects linked in Knowledge-Oriented Pathways), (Bizon et al., 2019) (Section 1.5.1) which is a database of biomedical knowledge organized as a knowledge graph and designed, in part, to support the elucidation of biological pathways of relevance to drug effects.

Although the systematic study of COPs has been lacking, a similar concept of '*Adverse Outcome Pathways*' (AOPs) has been commonly employed in regulatory chemical toxicology "to

support chemical risk assessment based on mechanistic reasoning (Organisation for Economic Cooperation and Development, 2020)." An AOP is defined as "a model that identifies the sequence of biochemical events required to produce a toxic effect when an organism is exposed to a substance".(of Health et al., 2021) Principally, AOP has served to formalize and standardize the description of the key processes that link a 'molecular initiating event' (MIE, i.e., the initial interaction of an exogenous molecule with an organism) with a downstream 'adverse outcome' (AO, i.e., the observable adverse response to the exogenous molecule at the whole-body level). The increased use of AOPs has enabled the rapid advancement of chemical toxicology research via more specific and detailed descriptions of toxic phenomena.(Tollefsen et al., 2014) Multiple examples of AOPs can be found in the AOP Wiki,(Society for Advancement of AOPs, 2021) and in the Organization for Economic Co-operation and Development (OECD) library.(Organisation for Economic Cooperation and Development, 2020)

Principally, AOPs have served to formalize and standardize the description of the key processes that link a 'molecular initiating event' (MIE; i.e., the initial interaction of an exogenous molecule with an organism) with a downstream 'adverse outcome' (AO) (i.e., the observable adverse response to the exogenous molecule at the whole-body level). The increased use of AOPs has enabled rapid advancement of chemical toxicology research via more specific and detailed descriptions of toxic phenomena.(Tollefsen et al., 2014) Multiple examples of AOPs can be found in the AOP Wiki,(Society for Advancement of AOPs, 2021) and the Organization for Economic Co-operation and Development (OECD) library.(Organisation for Economic Cooperation and Development, 2020)

A similar concept of '*Therapeutic Outcome Pathway*' (TOP) was discussed recently.(Morgan et al., 2016) It was introduced in a specific context as a means to help guide the development of personalized in vitro CSRA (chemosensitivity and resistance assays) platforms and discussed, along with AOP, as a general means to identify key microenvironment components and assay readouts/endpoints that could help in predicting therapeutic response and drug resistance. Another related concept of 'Clinical Pathway' (CPW) was discussed in the literature as well (Kinsman et al.,

2010) as a tool or framework to support evidence-based, organized clinical decision-making and effective patient care plan.(Kinsman et al., 2010)

We define COP by analogy to the AOPs, but in the broader context of systems chemical biology(Oprea et al., 2007) and clinical pharmacology as opposed to chemical toxicology. We formalize the COP as a sequence of biological events initiated by the exposure of a patient to a drug (a molecular initiating event) and leading, through a series of functionally linked perturbations at different levels of biological organization, to a specific, observable clinical outcome. By this definition, COP shares some elements with the above concepts of MOA, TOP and CPW but it is broader and more inclusive linking molecular, biochemical, and therapeutic events in a comprehensive pathway of drug action. Most importantly, as we discuss in the later part of this paper, COP can be elucidated computationally by mining biomedical knowledge graphs(Bizon et al., 2019) paving the way for instructive generation of novel drug discovery and repurposing hypotheses.

3.2.3 Defining Clinical Outcome Pathways

We define COP as a chain of functionally connected biological events with each element of the chain corresponding to a common term as defined in biomedical ontologies such as MeSH(Lipscomb, 2000) or MONDO(Mungall et al., 2017). Using AOP as an inspiration, three key elements comprise COPs: (1) *'Molecular Initiating Event'* (MIE) (2) *Intermediate Event(s)*; and (3) *Clinical Outcomes*. The three components are further defined as follows:

1. *'Molecular Initiating Event'* (MIE):
    (a) ligand-target interaction(s)
2. *Intermediate Event(s)* include a chain of some or all of functionally connected biological processes such as:
    (a) gene expression,
    (b) protein production,

(c) receptor signaling pathways,

(d) metabolic pathways,

(e) protein-protein interaction,

(f) cell-cell interaction, or

(g) tissue function

3. *Clinical Outcomes* such as:

(a) reduction in disease symptom(s),

(b) clinical pharmacology assays results showing normal values, or

(c) complete relapse of the disease.

We have provided a visualization of the Clinical Outcome Pathway elements in Figure 3.2. In this figure, we construct a COP as a graph like entity, where interactions begin and end. We must find a chain of events which enable a drug to find

To illustrate the concept, COPs for several drugs with their US Federal Drug Administration (FDA) approved indications are shown in Figure 3.3. Here we show three examples of FDA-approved drugs and the established information for how they interact with the human body. We see that COPs will naturally arise if we arrange the interacts as we have. These three pairs are drug and disease pairs which would be common to many pharmacists: (1) metformin and diabetes; (2) natalizumab and multiple sclerosis; and (3) diazepam and anxiety disorders. In the columns of the figure, we present a Clinical Outcome Pathway divided up into four components, the first column is the FDA approved compound, the second is the molecular initiating event, the third column is the chain of intermediate events, and the fourth column is the clinical outcome. Each of these columns are interconnected via a blue arrow, showing the flow of effect from each stage to the next. All three of the selected examples (diazepam, metformin, and natalizumab) demonstrate how well-established pharmacological profiles of drugs can be readily mapped onto the COPs framework.

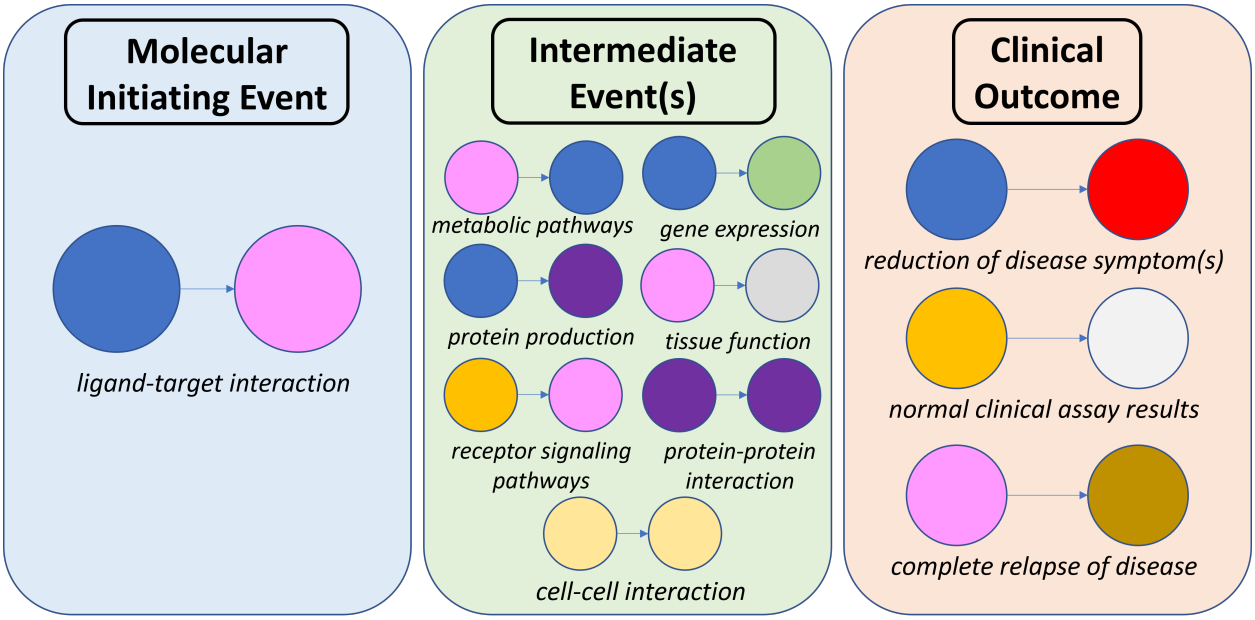


Figure 3.2: A visualization of the three pieces of the Clinical Outcome Pathway. The teal block on the left represents the MIE (*Molecular Initiating Event*). The green block in the middle represents the potential IEs (*Intermediate Events*). The orange block on the right represents the potential COs (*Clinical Outcomes*).

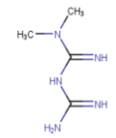
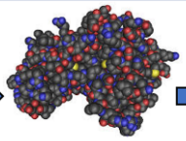
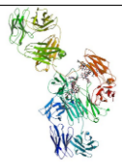
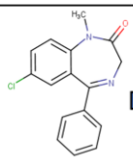
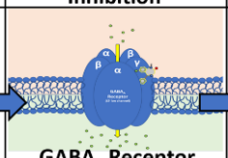
FDA Approved Drugs	Molecular Initiating Events	Intermediate Events	Clinical Outcomes
 Metformin	 AMPK Activation	1. Inhibition of glucagon-induced increase in cAMP concentration 2. Decreased gluconeogenesis in the liver	Decreased insulin tolerance, and better control of blood glucose levels in patients with type II diabetes mellitus
 Natalizumab	 α4-integrin Inhibition	1. Inhibition of α-4-integrin binding to VCAM receptors on epithelial cells of blood-brain barrier 2. Decreased migration of lymphocytes into CNS	Decreased adaptive neuroimmune response, which decreases symptoms and disease progression in patients with multiple sclerosis
 Diazepam	 $GABA_A$ Receptor Modulation	1. Increased response to GABAergic signaling in CNS 2. Enhanced hyperpolarization of CNS neurons leading to neuronal depression	Symptom reduction in patients with generalized anxiety disorder and/or panic disorder

Figure 3.3: Examples of COPs for three US FDA–approved drugs (metformin, natalizumab, and diazepam). COPs were elucidated using scientific literature that details the pharmacology of each drug.(Calcaterra and Barrow, 2014; Hutchinson, 2007; Rena et al., 2017)

3.2.4 Explanatory and Prospective Value of COP

We shall illustrate the COP concept using several pharmacotherapy case studies. For instance, two drugs that belong to the same pharmacological class often do not yield identical clinical outcomes. This case is exemplified by the functional selectivity of μ-opioid agonists, which may cause (potentially, lethal) respiratory depression in patients with chronic pain who are prescribed opioid analgesics. Recent studies(Pedersen et al., 2020) have demonstrated that functionally selective μ-opioid agonists are less likely to induce respiratory depression when compared to non-functionally selective μ-opioid agonists. These outcomes are observed presumably because the functionally selective agonists activate μ-opioid receptors without concomitantly stimulating β-arrestin signal transduction pathways, unlike the non-functionally selective agonists. In contrast to MOA based classification of μ-opioid agonists, COP highlights the underlying distinctions between drugs of the same class in the chain of events causing differential therapeutic outcomes (in this case, selective μ-opioid receptor agonism). Consideration of these distinctions is also essential for the study of poly-pharmacology, which is highly relevant to drug discovery and clinical applications such as the treatment of schizophrenia.(Roth et al., 2004)

Any drug may also have multiple COPs, with distinct therapeutic outcomes of relevance to different diseases, which effectively implies the popular concept of drug repurposing.(Blatt et al., 2014) Thus, validation of a novel COP hypothesis linking an approved drug to a new indication could drive innovation in drug repurposing. A recent real-world clinical observation involving patients infected with COVID-19 illustrates the importance of employing the COPs framework for supporting drug repurposing. Physicians observed that patients who had been infected with COVID-19 and also had a history of heartburn that was treated with the over-the-counter drug Pepcid, had a lower mortality rate when compared to similar patients who had not been previously treated with Pepcid.(Freedberg et al., 2020) To explain these observations, it was hypothesized that famotidine, which is the active pharmaceutical ingredient in Pepcid, might improve clinical outcomes of COVID-19 because it is an antagonist of histaminergic H2 receptors. It was proposed that binding of famotidine to H2 receptor leads to subsequent inhibition of mast cell activation in

human lung tissue, thus effectively blocking or diminishing the 'cytokine storm' that is believed to be a contributing factor to COVID-19–related mortality.(Malone et al., 2021) The authors of this hypothesis did not use the term COP, but effectively, the hypothesis can be described in the form of COP as follows:

$$\begin{aligned} &\textbf{Molecular Initiating Event:} \text{ famotidine } \rightarrow \text{ histaminergic } H_2 \text{ receptor antagonism } \rightarrow \\ &\textbf{Intermediate Events:} \text{ inhibition of mast cell activation } \rightarrow \text{ cytokine level reduction } \rightarrow \\ &\textbf{Clinical Outcome:} \text{ decreased COVID-19–related mortality} \end{aligned} \tag{3.2}$$

The divergence between the COPs for famotidine as a treatment for its approved indication, gastroesophageal reflux disease (GERD),(Sabesin et al., 1991) and its hypothesized indication, COVID-19 infection, is depicted in Figure 3.4.

We sought to explore how the flexible nature of COPs enable them to provide rich complex pathways through ontological space. We note that nowhere in the definition are we limiting ourselves to a singular pathway. It is possible and often very revealing to present multiple pathways together to the user. We present one such case in Figure 3.4. In this case we show how the drug Famotidine inhibits two different diseases. In the topmost pathway, we explore how the Famotidine works to treat gastroesophageal reflex disorder (GERD).(DrugBank, 2021) This process occurs when famotidine inhibits the gene HRH2. The inhibition of HRH2 then causes the parietal cells to decrease pump activity. This pump activity leads to an overall pH increase in the stomach lumen (the inner lining of the stomach). Finally, this increase of pH alleviates the symptoms of GERD. This presents one pathway for how Famotidine's interaction with HRH2 can help treat a disease.

On the bottom pathway of Figure 3.4 we explore how famotidine reduces COVID-19 morality.(Square, 2020) Similar to the top pathway, the molecular initiating event is that famotidine inhibits HRH2. But from that starting event, the overall sequence of events diverges drastically. The inhibition of HRH2 causes the mast cells in the body to decrease their degranulation (the response to wounding of the cell). This reduction in degranulation helps to reduce pulmonary edema (a

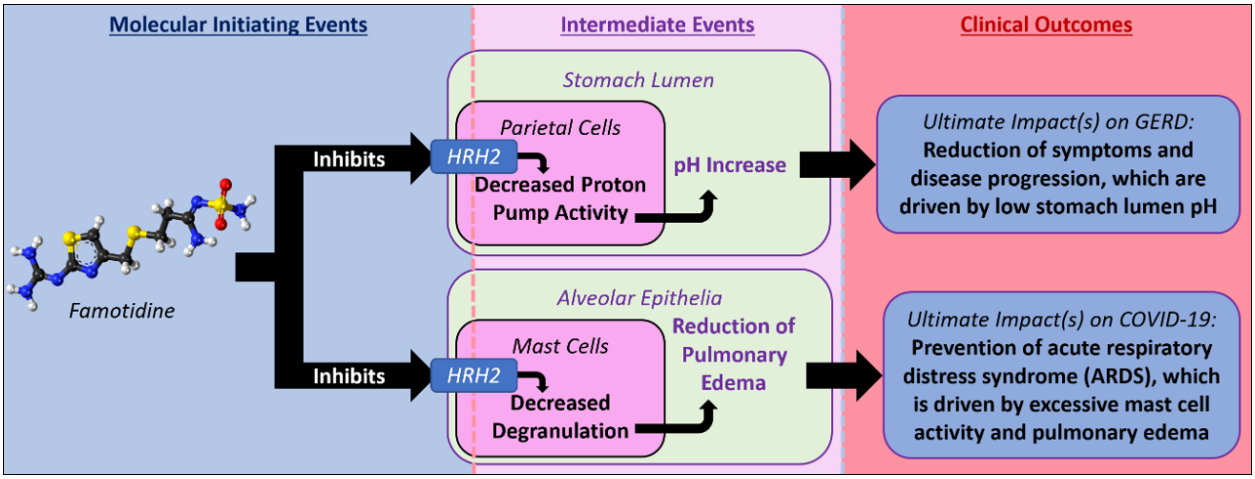


Figure 3.4: Example of two putative divergent COPs leading to two plausible clinical outcomes for the US FDA–approved drug famotidine: the top pathway corresponds to the approved indication for famotidine, gastroesophageal reflex disorder (GERD); the bottom pathway corresponds to a recently proposed hypothesis to explain the clinical observation that famotidine reduces COVID-19–related mortality and thus, can be repurposed as a treatment for this disease.

build-up of excessive lung fluid).(Liu et al., 2021) Ultimately, this process leads to the prevention of acute respiratory distress syndrome (ARDS), a condition where fluid leaks into the lungs. ARDS is a known serious complication of COVID-19 that is fatal, especially in the elderly or who have other conditions which raise their risk factor, such as diabetes and hypertension.(Gibson et al., 2020)

3.2.5 Computational Elucidation of COP

When investigating how drug (e.g., famotidine) application could lead to specific clinical outcomes (e.g., reduction of GERD symptoms or prevention of ARDS in patients with COVID-19) (See Figure 2), different aspects of processes that we collectively regard as COP can be searched for, which requires concordant analysis of diverse knowledge sources. These sources should include data on known biological drug targets, pathways, organs and tissues, disease phenotypes, and other biochemical entities related to pharmacology and pathophysiology of diseases. Furthermore, these sources should be integrated into a harmonized framework with standardized ontology to enable their efficient utilization.

For the last two decades, the size of the biomedical knowledge base has increased exponentially.(Lu, 2011) Efforts to consolidate this knowledge and represent it within a readily-interpretable user interface have yielded some success. Through knowledge extraction protocols and the development of robust ontologies, it is now possible to construct a high-order approximation of the current scope of biomedical knowledge that can be structured and stored in well-organized databases integrated into biomedical knowledge graphs.(Nicholson and Greene, 2020) Such graphs can be mined to explore potential COPs, as implemented in ROBOKOP(Bizon et al., 2019), providing insight into the pathways behind drug action. These elucidated COPs are theoretical upon conception, and therefore serve primarily as a means of hypotheses generation that can be validated in a clinical setting.

Mining of biomedical knowledge graphs can aid in drug discovery and repurposing efforts, as illustrated by the recent well-publicized use of a commercial knowledge graph to identify Baricitinib as a repurposed drug candidate for COVID-19(Richardson et al., 2020) and the discovery of drug combinations with synergistic and antagonistic against SARS-CoV-2.(Bobrowski et al., 2021) Notably, the relationships between various biochemical entities (drugs, genes, pathways, etc.) within COPs can be directly translated into respective graph representations, with biochemical entities as nodes and the functional relationships between them as edges that can be labeled by predicates describing the relationship between two specific nodes. Massive collections of biomedical data organized into knowledge graphs have been developed and curated in multiple research laboratories. These knowledge graphs can be queried to identify sub-graphs corresponding to COP. For example, scientific queries formulated in a plain language such as: "how does diazepam reduce anxiety in patients with a generalized anxiety disorder?" can be translated into a '*query pattern*' with the original query terms ("diazepam" and "anxiety") forming the flanking nodes of the subgraph and connected by fixed or arbitrary numbers of connected intermediary nodes. Such queries can be run against knowledge graphs such as those mentioned above to identify '*answer subgraphs*' with specific intermediary terms. Through this process, proposed COPs with key putative events may be matched with corresponding specific biological objects and validated

quickly and efficiently by exploring the supporting literature, thus providing a powerful hypothesis generation tool for drug discovery and repurposing. For modern knowledge graphs, the number of '*answer subgraphs*' for complex COPs can be massive; therefore, an emphasis on aggregation and ranking of these results is critical for future work. The creation of a common structured text format (e.g., XML files, JSON files) for COPs could promote the sharing and exchange of COPs into various databases and APIs for ranking. For instance, the publicly accessible portal ROBOKOP developed by our team enables the automated construction of COPs for drugs of interest.(Bizon et al., 2019) We expect that as knowledge graphs become more accessible, the generation of curated data-driven COPs would become a commonplace of computational biomedical research and discovery. Working towards this goal has been a key objective of the Translator Reasoner Application Programming Interface (TRAPI) initiative from NCATS.(National Center for Advancing Translational Sciences, 2021) Below we represent two case studies of ROBOKOP-enabled computational elucidation of COPs that can support drug repurposing.

3.2.6 Repurposing of Metformin for Chordoma

Metformin is an FDA-approved drug for managing type 2 diabetes.(Bailey, 2017) There have been several reports based on large scale observational and cohort studies suggesting its use for treating multiple diseases, including various cancers.(Pryor and Cabreiro, 2015) Using terms "metformin" and "cancer" to query ROBOKOP, we found that there were multiple links between this drug and chordoma, a rare type of bone cancer that usually develops in the bones of skull or spine.(Frezza et al., 2019) Figure 3.5A shows the relationships between the nodes in the answer graphs, and Figure 3.5B details multiple connections between metformin, genes, pathways, cells, and chordoma. As a prime example, we focused on the catalase (CAT) gene since it was related to only two types of cells (osteoblast and somatic cell). ROBOKOP provides linkages to published papers or other type of data that support edges between nodes in the answer graphs and enable interpretation of these graphs. Thus, Dai et al.(Dai et al., 2014) reported that metformin increases CAT activity in mice without increasing its expression. Another study(Schreurs et al.,

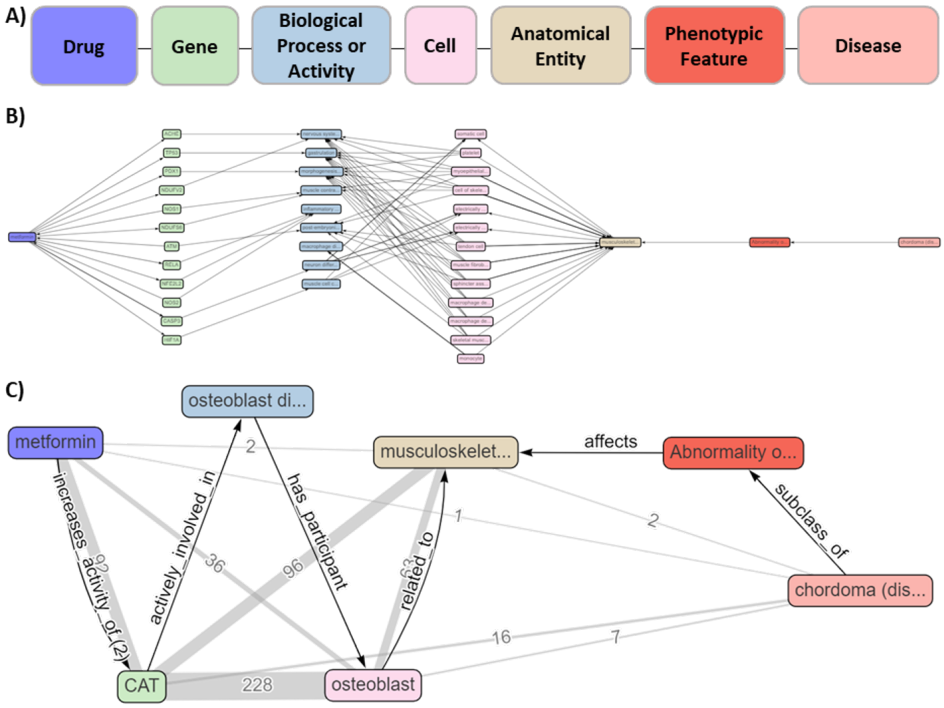


Figure 3.5: Possible COPs linking metformin to chordoma. A) Question graph for the metformin-chordoma COP; B) Complete knowledge graph of the ROBOKOP query showing multiple relationships between metformin, genes, biological pathways or activities, and cells to an anatomical entity, phenotypic feature, and chordoma. C) Putative COP of metformin and chordoma showing that metformin increases CAT activity, which is involved in osteoblast differentiation in the musculoskeletal system. Increased osteoblast differentiation has been shown to reduce abnormality in the musculoskeletal system, lowering the growth of sarcoma cells. To replicate this analysis, the user might go to `https://robokop.renci.org/` and select "Ask a quick question." Then, select "Template" and pick the first option "What is the COP for $name1$ and $disease1$?" Set the drug to "metformin," and the disease to "chordoma." The answered query can be found in this link: `https://bit.ly/3vFNzx6`.

2020) provided support for an edge between CAT and "osteoblast" nodes, reporting that ex-vivo osteoblast colony growth rate was 95% greater in transgenic mice with human mitochondrial CAT than in the wild-type mice. Lastly, an increase in proper osteoblast differentiation has been shown to reduce the development of chordoma cells since this type of tumor, along with other sarcomas, is characterized by uncontrolled proliferation and maintenance of undifferentiated osteoblasts.(Marie et al., 2015)

Metformin is currently in clinical trials for treating different types of sarcomas,(Longhi and Rizzoli, 2021; Molenaar et al., 2017) and similar reasoning could be put forward as a hypothesis

for metformin as potential treatment for chordoma as well. Figure 3.5C illustrates the underlying COP elucidated with the standard ROBOKOP query. Notably, the answer graph can be translated into a statement explaining the COP hypothesis: Metformin increases the activity of catalase that leads to proper osteoblast differentiation, potentially reducing the uncontrolled proliferation and maintenance of undifferentiated osteoblasts in sarcomas, including chordoma.

3.2.7 Imatinib – Gastrointestinal Stromal Tumor – Asthma Repurposing

Imatinib is an FDA-approved kinase inhibitor used to improve the prognosis of patients with chronic myeloid leukemia (CML).(Capdeville et al., 2002) Imatinib is also used to treat gastrointestinal stromal tumor (GIST).(Lopes and Bacchi, 2010) In 2017, a clinical trial demonstrated that imatinib was effective in treating severe asthma.(Cahill et al., 2017b) Imatinib was first identified as a selective inhibitor of BCR-ABL kinase.(Capdeville et al., 2002) It was later found that this drug binds to several kinases, including KIT.(Buchdunger et al., 2000) Here, we employed ROBOKOP to elucidate the COPs of imatinib that involve KIT inhibition and result in treating CML, GIST, and asthma. This process is vital and involved in three biological pathways identified in ROBOKOP: multicellular organism development, hemopoiesis, and erythrocyte differentiation. Figure 3.6A shows the COP linking imatinib and CML. As the first **MIE**, imatinib inhibits the activity of KIT,(Buchdunger et al., 2000) which reduces megakaryocyte-erythroid progenitor cells in the bone marrow. As a result, imatinib inhibits nuclei proliferative signals, which induce leukemic cell apoptosis. Figure 3.6B shows the COP of imatinib and GIST. The majority of KIT proto-oncogene mutations found in GISTs patients induce constitutive kinase activation, inhibiting apoptosis and stimulating cancerous cell proliferation.(Lee et al., 2006) The inhibition of KIT in the enteric smooth muscle cell GIST patients induces cell apoptosis.(Lee et al., 2006) Figure 3.6C shows the COP relating imatinib and asthma. KIT is also present in lung mast cells and was hypothesized as a basis of the pathobiology of severe refractory asthma,(Fuehrer et al., 2009) which is characterized by an adverse response to traditional glucocorticoid asthma treatment.(Cahill et al., 2017b) Imatinib reduces airway hyperresponsiveness, a physiological marker of severe asthma, and airway mast-cell

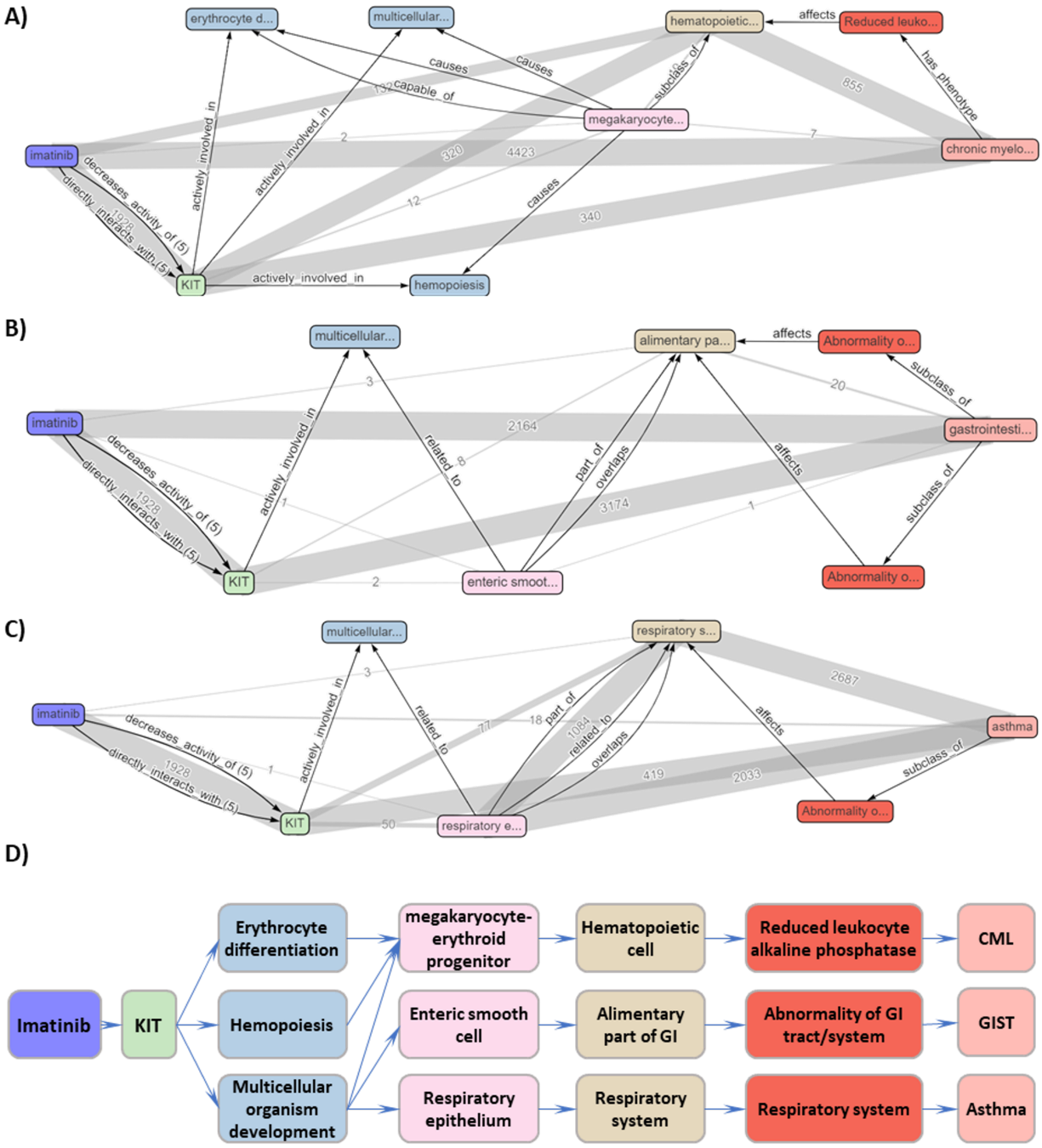


Figure 3.6: COPs for imatinib CML, GIST, and asthma involving KIT. A) Imatinib and CML; B) Imatinib and GIST; C) Imatinib and asthma; D) Integrated COP. To replicate this analysis, the user might go to `https://robokop.renci.org/` and select "Ask a quick question." Then, click on "Template" and select "What is the COP for $name1$ and $name2$?" The answered query can be found in these links: A) `https://bit.ly/3fXrGTm` B) `https://bit.ly/2TnGQcH` C) `https://bit.ly/3pbl7AP`.

numbers and activation in patients with severe asthma.(Cahill et al., 2017a) Integrated COP for Imatinib – KIT – . . . – CML/GIST/Asthma is shown in Figure 3.6D.

3.2.8 COPS - Final Remarks

We have defined the COP as a chain of key events [MIE -¿ Intermediate Event(s) -¿ Clinical Outcome] linked in mechanistic pathways that underly therapeutic outcomes of drug action. We presented several examples to illustrate both manual and automated elucidation of COP in characterizing drug-disease relationships and drug repurposing potential. We posit that the formalized structure of COPs, coupled with their elucidation via mining of large biomedical knowledge graphs, will deepen our understanding of the biological pathways of drug action and facilitate the generation of testable hypotheses to accelerate and advance drug discovery and repurposing efforts.

3.3 Semantic Query Patterns

3.3.1 Introduction

Knowledge graphs (KGs) represent real-world facts using nodes to describe ontological entities and edges to describe relationships linking those entities. The elementary unit of a KG is the `(node,edge, node)` triple structure. This representation provides straightforward modeling of domain-specific knowledge, such as how biological concepts interact with each other, or how different individuals relate to each other. An example in the biomedical domain would be `(influenza,correlated with,chills)`; taken from the biomedical knowledge graph ROBOKOP (Bizon et al., 2019).

The construction of KGs is enabled by domain-specific ontology. An ontology is a formalized specification that describes sets of entities in a specific way that places them into a specific category (Ehrlinger and Wöß, 2016; Feilmayr and Wöß, 2016; Uschold and King, 1995). Ontologies will have different levels of complexity based upon their described purpose; as an example, the MeSH ontology (Lipscomb, 2000) is used to tag papers when they are submitted to

academic publications, as this is a very broad range of topics it must cover, this ontology often trades granularity for generality. One example is the term `D007501`, which simply represents the concept of *Iron*. Another ontology, such as *Chemical Entities of Biological Interest (ChEBI)* (Hastings et al., 2016), aims to capture nuanced details of chemical compounds. In this ontology we can locate the entities: `CHEBI:82664`, `CHEBI:29033`, and `CHEBI:29034`. Each of these entities represents a slightly different atomic form of iron atoms with a different electrical charge. Such detailed representation would be unnecessary for citations in papers, but is very necessary in representing chemical space. The ontologies that a knowledge graph chooses to use to represent its nodes and edges can drastically affect what it represents well and what it represents poorly, which depends greatly on the goals and use cases of the graph. Some choices may lead to an overly granular representation that is difficult to explore, or an overly general representation that lacks detail (Uschold and King, 1995).

Furthermore, knowledge graphs provide extraordinary utility towards the generation of new knowledge from the analysis of existing knowledge. This process, called *hypothesis generation* (Gettys and Fisher, 1979), enables the generation of potential (node, edge, node) triples, which while not presently in a knowledge graph, have a high likelihood of being true. This process is also commonly referred to as *KG-mining* (Paulheim, 2017). These hypotheses enable domain experts to direct their exploration of new ideas from the domain of all possible hypotheses to those that are likely given the current evidence in a field.

Unfortunately, due to the vast amount of knowledge the scientific community has accumulated, modern KGs tend to be extremely large and complex, making it difficult to extract meaningful hypotheses (Ronfeldt and Arquilla, 2020). These graphs gain their power from the number and breadth of their connections, often with the number of nodes and edges in these graphs numbering in the billions. Therefore, when exploring how specific nodes in such a graph are connected, the results of even simple meta-path queries may contain millions of potential pathways, many of which may not contribute to the specific case an expert is exploring. Work discussing these spurious pathways can be found in relation to the over representation of hub nodes (Jain et al., 2021).

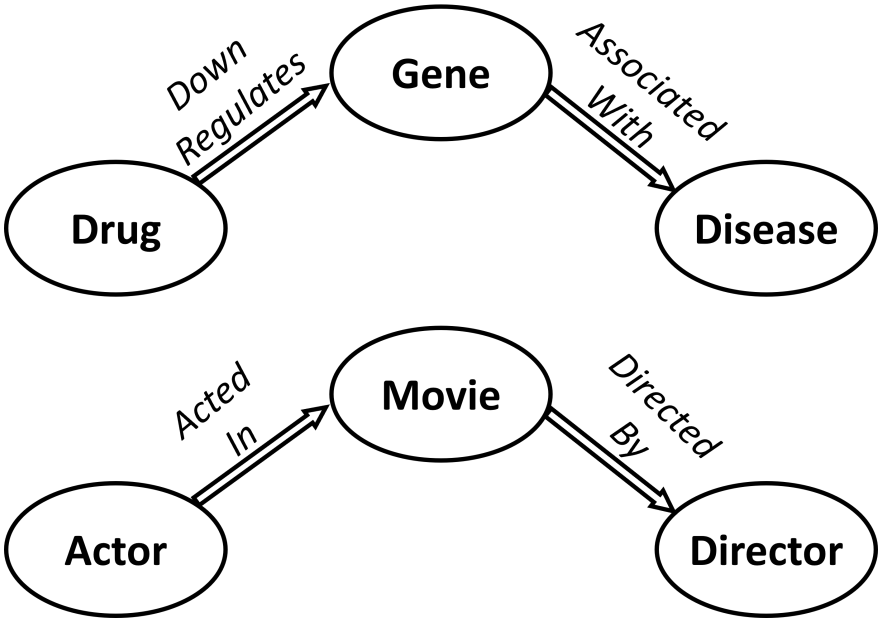


Figure 3.7: An example of two semantic pathways. On the top is a semantic pathways linking a drug and a disease; through some genetic factor. On the bottom is a semantic pathway for capturing actors and directors that have collaborated in a film.

Meta-paths are collections of node and edge labels that can express high levels of meaning when applied to a knowledge graph. We see an example of these pathways in Figure 3.7. In this figure, we see two pathways both with the same superficial structure (three nodes forming a simple chain), but the labels and, thus, the information that can be understood from using these pathways is very different. The top pathway represents a biological meta-path, while the bottom pathway is relevant to film. This represents how these pathways are contextual, and the importance of what graphs they will query. In knowledge graphs, nodes represent real life concepts and the ontological classification of these concepts. An example of a node from a knowledge graph may be **aspirin**, which would have the label *Chemical*. However, in *ontological space* the nodes are "concept labels", such as *Movie*, *Chemical*, or *Gene*. The edges in *ontological space* are relationships that describe how two concepts may relate to each other. When viewed in this way, these meta-paths capture how categories of objects can relate to each other. These can be used to extract information; we see an example of this in the bottom pattern of Figure 3.7, which when queried against a knowledge graph of films, will produce a list of actors and directors that have collaborated together on movies. Once

these collections of node and edge labels have been generated, pattern matching processes can be used to find matching patterns in the graph (Gallagher, 2006). In previous studies, these collections of node and edge labels have been referred to as "query patterns" and "meta-paths" (Dong et al., 2017; Binder et al., 2022).

An ideal case of hypothesis generation in knowledge graphs would entail an expert carefully looking at all possible pathways that link different entities in a graph and deciding which pathways have potential of leading to a novel discovery. Unfortunately, this would be impossibly labor-intensive as the number of potential linkages in a graph could be in the millions (Alves et al., 2022). Thus,kms62kms the majority of approaches to knowledge graph mining rely upon automated mining using statistical methods, such as embedding methods like Node2Vec(Grover and Leskovec, 2016) and TranSE (Wang et al., 2014). Machine learning approaches attempt to automate the process of discovery without expert curation, sifting through the billions of potential pathways in a knowledge graph and providing those of interest to a user. It is difficult to validate the results produced by these methods. Additionally, the state of *possible* meta-paths is exponential in the size of the node labels, and is much larger than those with true semantic meaning, making the likelihood of finding semantically meaningful meta-paths low.

Expert constructed meta-paths serve as a heuristic methods for hypothesis generation, in which specific combinations of node and edge labels are deemed as predictive and informative by expert users. One such example of this type of meta-path comes from the Clinical Outcome Pathway (COP) methodology (Capuzzi et al., 2018; Korn et al., 2022), which aimed to categorize the biological linkages between *drugs* and *disease* (COPs are discussed in more detail in Section 3.3.2). An issue with these meta-paths is they can only be applied to a specific knowledge graph for one particular use case. Presently the only way to find these semantic pathways is with the meticulous work of domain experts. All possible pathways that may cause two concepts to relate must be documented in the formalization of these paths, such as the case with COPs and AOPs. Once completed, these robust formalizations must be shared with the broader community. The extensive time committed to the construction of these pathways creates significant restrictions.

When developing meta-pathways, experts have two options for constructing them. The first option is to develop a generic pathway, i.e., one that is not built to conform to any one ontology, but instead uses general concepts; this is the case with COPs. When a user wants to query a graph with this pathway, they must first build a mapping from the pathway concepts to the ontology of the target knowledge graph, which may not capture ideas or concepts they presented and is time consuming. The second option is to tailor the semantic pathway to a specific ontology and specific knowledge graphs. In this case, if the underlying graph is changed or experts seek to search other databases, the patterns that the expert spent so long developing may no longer be valuable. In either case, the significant time investment is undesirable.

We seek to implement an approach in which biomedical experts work iteratively with a KG mining algorithm to generate, evaluate, and find meta-paths. This is because these pathway are highly contextual to the complex nature of knowledge graphs. One of the main advances of the COP methodology is to acknowledge this inherent complexity of the real world and attempt to capture it. Previous work has looked at the "semantic subgraph", a *domain/task oriented* regular-path expression (Hou et al., 2022). We seek to expand upon this definition, looking to define these expressions as an area of all possible paths; this is discussed at length in Section 3.3.4. We title the patterns that our approach generates **Semantic Query Patterns** (SQPs). Our method of developing these SQPs can be viewed as a *ping-pong*, where in the user attempts to find a pathway, and then receives insight and feedback, and creates more complex pathways in response to this feedback. By having expert users generate these SQPs for the graph, we enable more complex and dynamic pathways to be generated. The iterative approach in the hands of an expert provides us an avenue to apply our frameworks to novel problems and domains.

In our survey of the literature, we have found no other group that has used the term **semantic query pattern** in a knowledge graph context; although many groups have adopted similar terminology. So we feel we may use this terminology to express our conception. We have provided a summary of works and the similar terms in the related work section (Section 3.3.1).

In our algorithmic work, we aim to extend our existing CompactWalks methodology, with the creation of the *SQP-Hunter* Algorithm (Algorithm 4), which enables domain experts to find SQPs quickly and efficiently from a set of training data. The ultimate objective is to find a regular pattern that gains high semantic similarity for many expert provided pairs. By choosing meaningful pairs of nodes and well constructed knowledge graphs we claim that these regular patterns also have semantic meaning.

To evaluate the SQPs, an evaluation metric is necessary. The recent publication of the CompactWalks framework (Hou et al., 2022) (covered in Section 3.3.2) provides us a strong opportunity to meaningfully evaluate these semantic queries. Instead of attempting to evaluate a singular pathway, the CompactWalks methodology uses subsampling of a graph to evaluates entire families of regular languages.

Our contributions are as follows.

- We have formalized the concept of semantic query patterns (SQPs), and create a theoretical framework to fit these queries into a more generalized view of graph queries.
- We have constructed an algorithm that, using the CompactWalks framework (Hou et al., 2022), enables expert users to generate SQPs for specific problem domains.
- We implemented this algorithm in the form of a web application that we both host and have made open source so individuals may host their own versions.
- We tasked expert users with the creation of task-specific SQPs in the field of drug discovery and disease research. We then explored the resulting top-performing pathways.

Chapter outline. We discuss related work in Section 3.3.1. Section 3.3.2 presents the preliminaries for our work. Section 3.3.3 and our research tasks and hypotheses. Section 3.3.4 introduces our proposed approaches. Section 3.3.5 details the implementation of the proposed approach, the case studies we performed, and the results of these case studies. In Section 3.3.6 we discuss the limitations of our approach. Finally, Section 3.3.7 concludes the paper.

Related Work

In 1979 Gettys and Fischerwork defined a broad theoretical model for the task of hypothesis generation (Gettys and Fisher, 1979). They concerned themselves with issues of gathering all possible hypotheses, statically pruning invalid theories, and observing outcomes. They also performed psychological studies in which individuals were presented data and asked to come up with hypotheses. As the authors stated "[the hypothesis generation] model is to be applied in those situations in which the decision maker is attempting to generate hypotheses that will account for the available data... During the actual experiment the subjects were told that the generation of a 'correct' hypothesis was not nearly as important as the generation of new hypotheses that were plausible in the light of the data." We are inspired by this inital work in hypothesis generation in our pursuit of it here in regards to knowledge graphs.

In work by Angles et al., they categorized and defined a broad expressive terminology and conceptualization for the task of querying graph databases (Angles et al., 2017). This work specifically focuses on query languages for these graph databases. It included a large discussion how all the disparate graph query languages (such as CYPHER (Team, 2022), SPARQL (Harris et al., 2013), and Gremlin (Rodriguez, 2015)) all share the same underlying primitives. This work covers how the described primitives could construct more complex queries such as those over a path, and how to harmonize the representation into each graph query language.

There exists a breadth of work on completing the querying of graph databases for complex queries (Barceló, 2013; Barceló et al., 2014). Specific work has been expanded upon the question of query completion, to explore the issue of querying regular patterns over graph databases. Specific emphasis placed upon runtime and memory concerns in some of the work, showing theoretical concerns for different query constructions.(Barceló et al., 2014). These works completes a suite of proofs regarding the asymptotic complexity of querying these graphs for different forms of queries.

Liang et al. covered matching semantic patterns in Heterogeneous Information Networks (HINs) (Liang et al., 2016). This work covered the specific problem of weighted directional HINs, where nodes represent individuals or forms of communication and edges represent relationships

between individuals. This work introduces PROphetic HEuristic Algorithm for Path Searching (PRO-HEAPS), which uses a pre-computation of a heuristic network and modified A^* search to find the Top-k shortest paths connecting two nodes. This work differs from our own because: (1) it uses a weighted graph, while most knowledge graphs are unweighted and (2) it assumes a non-trivial query (in the work called a "Prophet Graph") has already been provided, while we explore the case where a query does not exist. Further information and exploration of HINs can be found in the work of Shi et al. (Shi et al., 2017).

The similar term "semantic graph pattern" has appeared in existing literature. Work by Zheng et al. describes a semantic graph pattern as "a set of structures that convey equivalent semantic meanings" (Zheng et al., 2016). In their work, they look at schema-free RDF datasets. These semantic graph patterns occur when two distinct queries produce the same result and therefore have equivalent meaning in the graph. They produce algorithms for mining these queries and providing users the ability to run a single semantic query that runs multiple semantically equivalent database queries. They do this through "semantic graph edit distance," which is calculated by taking the cotopic distance between two ontological types, although it is unclear if this is the actual "edit distance" of the string representation of labels. This method assumes that the difference between any two selected node labels or edge labels can be calculated with a numerical score, which our method does not require.

The term semantic graph patterns has seen some use in biomedical literature in the past. Vogt says "[semantic graph patterns when applied consistently throughout a data repository that stores and manages phenotype descriptions, the set of templates would specify a semantic model for phenotype data and metadata (Vogt, 2021)." Other biomedical works have also touched on this issue. Literature by Bakal et al. (Bakal et al., 2018) explored the question of "semantic patterns" and "semantic paths" over the SemMedDB (Kilicoglu et al., 2012) repository. We also see work done with semantic queries in for the use in analysing the data of the elderly, collecting sensor data from them in a network of sensors and selectively querying that data to recieve insights about the patients condition (Culmone et al., 2014).

We see the term "Semantic Query Pattern" used explicitly by Franke et al. in their work on a tool for B2B transactions. In this tool a user is aided in the construction of SPARQL queries (Franke et al., 2018). The work shows no images of the tool in a functional state and the links provided to the tool are no longer active. This work also appears in a different domain than the field of algorithmic based KG research.

Work by Čebirić et al. covers the idea of semantic graphs coming from RDF data, the task of summarizing of these database (Čebirić et al., 2019). The work involves summarizing them through reduction of information, identifying isomorphisms, linking existing graphs. It also builds out a complex taxonomy of inputs, outputs, and methodology for grouping classifying existing algorithms. These summarization methods aim to cover any RDF graph.

3.3.2 Preliminaries

Meta-paths

Meta-paths are collections of node and edge labels that provide information on how object types are related (Sun and Han, 2013). The classic examples of meta-paths provided by Sun and Han is the "*author-paper-author*" path and the "*author-paper-venue-paper-author*" path, which provides connections between authors who published a paper together or who published in the same venue, respectively (Sun and Han, 2013). We have also shown two examples of meta-paths in Figure 3.7, the first of which is "*drug-gene-disease*," and the second is "*actor-movie-director*". These meta-paths when pattern matched against a knowledge graph can provide specific instances of nodes and edges that match the labels within and may give further insight into the interaction of those particular nodes. Meta-path mining has enabled the extraction of relationships and features of specific classes of objects from knowledge graphs (Kong et al., 2012; Shi and Weninger, 2016; Wang et al., 2019), and serve as the foundation of the mMtapath2vec embedding framework (Dong et al., 2017).

Specific domain and task meta-path creation frameworks have been created in recent years, particularly in the biomedical domain. The two most prominent examples are *COPs* (clinical

outcome pathways) (Capuzzi et al., 2018; Korn et al., 2022) and *AOPs* (adverse outcome pathways) (Tollefsen et al., 2014; Wittwehr et al., 2017). COPs describe the therapeutic action of a drug by relating a molecular initiating event (MIE) caused by a drug to a series of key event (KE) of increasing biological scale, which culminate in an observed clinical outcome (CO). COPs were developed to capture the complex relationships between drugs and diseases, attempting to develop a more robust model than the existing *mechanism of action* framework, which implicitly assumes all causes of drug-disease relationships could be captured through a singular biological mechanism, which is oftentimes too simplistic to capture the full interaction. COPs also enable the querying of knowledge graphs in these specific cases for hypothesis generation. AOPs were created by the toxicology community to document how pollutants in an environment may be detrimental to the health of humans, the pathways follow along the chain of how toxic agents are ingested, which biological systems they interfere with, and how interference in those systems may damage human health.

CompactWalks

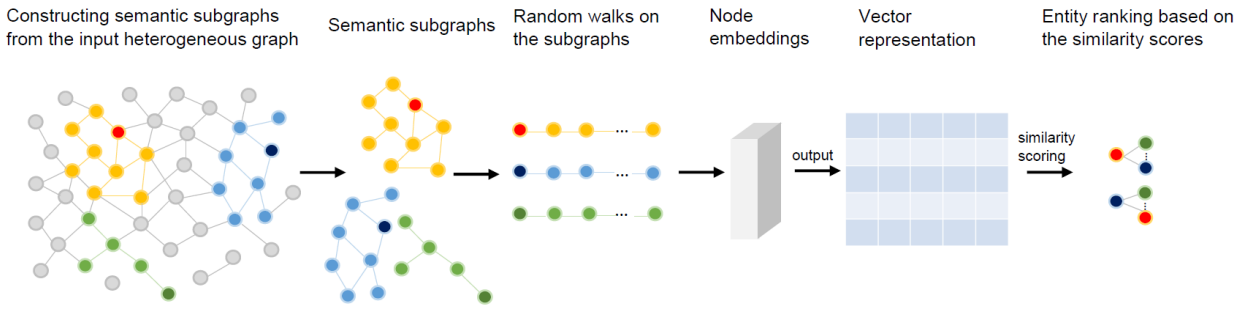


Figure 3.8: A visualization of the CompactWalks methodology. Reproduced with permission from Hou 2022 (Hou et al., 2022).

Graph embedding is the process of transforming nodes and edges of a graph and placing them into a continuous vector space (Goyal and Ferrara, 2018). Capturing the highly diverse and complex relationships present in a graph structure in an array of floating point numbers is a significant challenge, both conceptually and mathematically. Nodes in graphs can exist in incredibly complex local environments, and every neighbor of a node also exists in its own unique and complex

local environment. Graph embedding algorithms must find a way to preserve this local complexity of how nodes in a graph relate to one another. Once these embeddings have been created, the embeddings of different nodes in the graph may be used to compute a similarity metric between the values, such as *euclidean* or *cosine*. Many approaches have been constructed to perform such embeddings of nodes in a graph, e.g., Node2Vec (Grover and Leskovec, 2016), Metapath2Vec (Dong et al., 2017), and CompactWalks (Hou et al., 2022).

The CompactWalks framework by Hou et al. specifically aims to tackle the problem of evaluating the performance of queries on large knowledge graphs (Hou et al., 2022). This work expanded on the previous Metapath2Vec frameworks (Dong et al., 2017), but introduced the idea of regular languages as a way of expressing complex queries in an efficient manner. By restricting the graph and subsampling to only the regions of the graph expressed by the regular language, this approach aimed to only utilize relevant areas of the graph in the generated embeddings. Ultimately this methodology proved successful, enabling the creation of embeddings on the ROBOKOP knowledge graph, with over 10 million nodes and 100 million edges.

By limiting the how the graph is explored in the random walk stage, CompactWalks decomposes the problem of embedding into two distinct phases: (1) pruning of the source graph, and (2) embedding the pruned graph. Phase (1) is enabled by a family of regular languages that defines which pathways through the graph are valid. In phase (2), the filtered graph is then fed into any desired embedding algorithm as an input. This methodology enables the CompactWalks framework to operate on extremely large graphs with hundreds of millions of nodes. First, the regular language family is translated into a graph database query language such as Cypher. Then, the source graph is queried, leveraging the existing capabilities of graph databases, such as Neo4j, to quickly find all pathways through a very large graph. The resulting pathways are then stitched together into the desired subgraph. For two target nodes, the random walks can be joined together to create a singular data set that can then be fed into a skip-gram model (Mikolov et al., 2013) (a specific structure of neural network) to generate embeddings. We can then use a distance measure to compare how closely related two nodes are over the space of an entire knowledge graph.

3.3.3 Research Tasks and Hypotheses

Our overall **research goal** is to make the task of constructing domain-specific regular languages that, when used to filter or query on knowledge graphs (KGs), will provide unique insight into how particular sets of objects within that domain are connected to each other. We put forth a **general hypothesis** that for any domain-specific knowledge graph, with a set of pairs of objects that have some meaningful relationship, you could construct a meta-path that describes the relationship between these objects, and that a domain expert will regard as enriching in the pursuit of new knowledge.

In the context of our analysis, our **main research task** is to produce a family of regular languages $\mathcal{L}$; the inputs to our methodology are an embedding methodology $\mathcal{E}$, a distance measure d, a set of positive data pairs P^+, and a set of negative data pairs P^-. We will maximize similarity scores for the embeddings of nodes restricted by a regular language in positive pairs, and will minimize similarity scores for those embeddings of nodes in negative pairs. That is, the language well suited towards similarly embedding related pairs in that domain while separating those who are not meaningfully related. We call this regular language family a **semantic query pattern**.

We formalize the main research task as follows: Let N be a set of nodes from a KG G, and $\mathcal{E}$ be an embedding tool that provides an embedding vector for $n \in N$. We score the relationship of two nodes $n_1, n_2 \in N$ using the distance function:

$$\tau(n_1, n_2) = d(\mathcal{E}(n_1), \mathcal{E}(n_2)) \tag{3.3}$$

The result of ranking these scores (higher rank is better) over sets of nodes is presented to the domain expert. Let $P^+ = \{(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)\}$ and $P^- = \{(\hat{a}_1, \hat{b}_1), (\hat{a}_2, \hat{b}_2), \ldots, (\hat{a}_m, \hat{b}_m)\}$. We seek to achieve high similarity for positive pairs $\tau(a_1, b_1)$, $\tau(a_2, b_2)$, …, $\tau(a_m, b_m)$, while achieving low similarity for negative pairs $\tau(\hat{a}_1, \hat{b}_1)$, $\tau(\hat{a}_2, \hat{b}_2)$, …, $\tau(\hat{a}_m, \hat{b}_m)$.

Our **specific hypothesis** is as follows: domain experts with a KG are able to construct an semantic query pattern (defined in the next section, Definition 3.8) that prioritizes the discrimination of semantically related pairs of nodes while ignoring those relationships that are not meaningful. Moreover, this pathway will be fact rich and lead to novel assertions. That is, when the produced regular language family $\mathcal{L}$ is used as an input to the CompactWalks system, the embedding vectors produced for the positive pairs will have a high similarity score, while those embedding vectors of negative pairs will have low similarity score. In this way the language will lead to the separation of truly associated objects in the domain from objects that have been stated to not be related in any way.

We will discuss the issue of pair selection and the specific problem domain of drug discovery in Section 3.3.5.

3.3.4 Semantic Query Approach

Experts often have *intuition* that is very difficult to put into words. *Intuition* can be defined as an underlying understanding of processes that utilizes the ability of the human mind to find patterns that connect entities, but one is not necessarily able to explicitly verbalize these patterns (Kuhn, 1989; Greenhalgh, 2002). As an example, a biomedical expert may have the ability to identify that two genes operate extremely similarly in the human body, but may have no explicit reason to explain this association; in this case, through decades of experience and study the expert has an intuitive understanding of the field of genetics.

We seek to help experts explore and explain these underlying intuitive patterns and to help them formalize these ideas into meta-paths over a knowledge graph. Using a collection of similar node pairs gathered through intuitive exploration as input to our approach, we seek to explore the semantic space and uncover an SQP that can provide mathematically describe these intuitive patterns. With the aid of a domain-expert, these pathways may uncover the underlying mechanisms that link objects in a domain. Using technology to bridge the gap between intuitive and explainable,

our tool can aid in finding and making explicit the intuitive metric of similarity the expert used when compiling their list of similar node pairs.

We propose a methodology that semi-automatically generates a regular language $\mathcal{L}$ for a family of pathways through G that has semantic meaning for the subject matter of our knowledge graph. Capturing semantic meaning is an exceedingly difficult task, so we include input from expert users in our pathway generation task.

The objectives we seek to optimize for in this regular language $\mathcal{L}$ are as follows:

1. The returned semantic language $\mathcal{L}$ has expression length k.

2. The returned semantic language $\mathcal{L}$ maximizes similarity scores in the embedding approach $\mathcal{E}$ for P^+, the expert provided positive pairs.

3. The returned semantic language $\mathcal{L}$ minimizes similarity scores in the embedding approach $\mathcal{E}$ for P^-, the expert provided negative pairs.

This methodology works through expert interference. We undertake the task of automatically generating semantic query patterns. Our approach is data-driven; we call upon domain experts to provide positive and negative examples of relationships, and we leverage existing work on knowledge graphs to analyze these examples. Our goal is to automate the generation of a semantic language $\mathcal{L}$, which can be tuned to various problems by simply changing what data we analyze. If we analyze a biomedical knowledge graph G, then inputting a set of drugs known to treat diseases should produce a COP; while inputting a set of toxins known to cause diseases should produce an AOP. Refer back to our explanations of COPs and AOPs in Sections 3.2 and 3.3.2.

Definitions

First, we must provide a formal definition of a graph.

Definition 3.1. Graph.

We define a graph as $G = (V, E)$, where V is a collection of vertices, or nodes, and $E \subseteq V \times V$ is a collection of edges linking vertices. We use $|V|$ and $|E|$ to denote the number of nodes and number of edges in G, respectively.

We need to extend this definition to a knowledge graph, that is, a graphical structure used to capture information. For example, ROBOKOP (Bizon et al., 2019) and HetioNet (Himmelstein et al., 2017) are two knowledge graphs in the biomedical domain.

Definition 3.2. Knowledge Graph. We define a knowledge graph as $K = ((V, E), T_V, T_E, \phi, \psi)$, where (V, E) is a graph, T_V is a set of node labels, and T_E is a set of edge labels. A knowledge graph K must have a node type mapping function $\phi : V \to T_V$ and an edge type mapping function $\psi : E \to T_E$.[1] ϕ must be defined for each node $v \in V$ and ψ must be defined for each edge $e \in E$. $|T_V|$ denotes the number of node labels for G and $|T_E|$ denotes the number of edge labels for G.

K is *homogeneous* if $|T_V| = 1$ and $|T_E| = 1$.

K is *heterogeneous* if $|T_V| + |T_E| > 2$.

We must also formally define a **path** within the graph. These pathways are strings of nodes, and often serve as the results of queries made on the graphs.

Definition 3.3. Path. Let G be a graph. Let p be an ordered list of vertices of G. We require paths to have at least three elements because a single-element list would just be a node, and a two-element list would be a single edge. We say that a path p has length k if $|p| = k$; moreover, we can represent p as $\{v_1, v_2, ..., v_k\}$. We say that p is a *path* over G *iff*

$$\forall i \in \{1, k\} : v_i \in V \text{ and } \forall i \in \{1, k-1\} : (v_i, v_{i+1}) \in E$$

That is, p is a path over G if and only if all pairs of adjacent vertices in p have a corresponding edge in the graph G.

1. This definition of knowledge graph assumes a mapping from an edge to a singular edge label, this definition can be easily expanded to account for multi-label relationships.

We say $\mathcal{P}_G \subset \bigcup_{i=3}^{\infty} V^i$ is the set of all possible paths in G. More formally, $\mathcal{P}_G = \{p \in \bigcup_{i=3}^{\infty} V^i | \ p \text{ is a path in G}\}$.

Here, we seek to position the possible **ontological space** answers may take. This is the combination of node and edge **labels** that may make up any path from the graph. Such a space encapsulates all the ways any two objects *could* ever be connected.

Definition 3.4. Ontological space. The **ontological space** is all possible patterns that may be made up by different combinations of node labels T_V and edge labels T_E. The motivation here is to define the space in which all possible meta-paths are fully represented. We more formally describe it below.

Let L_k be a set of node and edge labels such that

$$L_k = (T_V \times T_E)^k \times T_V$$

That is, L_k is the set of all possible meta-paths with k node labels and $k-1$ edge labels.

Combining the families L_k for all possible values of k gives us the full possible meta-paths in G, $\mathcal{O}_G$. Where

$$\mathcal{O}_G = \bigcup_{k=3}^{\infty} L_k$$

This $\mathcal{O}_G$ provides us a useful formalization of the space in which meta-paths exist.

Here we formalize the concept of an **ontological reduction function**. This function reduces a specific instance of a **path** through a KG from it's particular nodes and edges to the ontological labels for those nodes and edges. This function acts a transition from the "real" world to the "ontolgical" world. As an example a path may be $p =$ (**aspirin**) $-$*inhibits* $\rightarrow$ (**prostaglandin synthesis**) $-$*causes* $\rightarrow$ (**inflammation**). This path summarizes the effect of aspirin on the condition of inflammation. And the reduction of this function would be $\delta(p) =$ {**drug**,*inhibits*,**biological process**,*causes*,**phenotype**}, this reduction broadly categorizes the effects described by the path and the concepts they represent.

Definition 3.5. Ontological reduction function. Let us define a function

$$\delta : \mathcal{P}_G, \phi, \psi \rightarrow \mathcal{O}_G$$

Where δ takes as input a path (see *Definition* 3.3), a *node label* function ϕ, and a *relation label* function ψ (see *Definition* 3.2). We then find the node and edge label for each component of the path. This function can be viewed as a *reduction* over any path $p \in \mathcal{P}_G$, from specific nodes into an ontological representation. We represent this as $\delta(p, \phi, \psi) = \{\phi(v_1), \psi(e_1), \phi(v_2), \ldots, \psi(e_{k-1}), \phi(v_k)\}$.

Here we seek to generalize the idea of a query. Taking inspiration from ideas seen in finite automata and Turing machines. We seek to represent how a user may **query** a graph more broadly as a selection over possible paths in the graph. We formalize this idea as deconstructing the problem into a function that must be able to take all possible paths from the knowledge graphs as input and either accept or reject that path. We then may look at all possible paths in the graph and find set of paths that would all be accepted.

We seek to present a way to represent if any path taken from the all possible paths in a knowledge graph is interesting or useful for a particular path of exploration.

Definition 3.6. Query. Queries are filters for paths on knowledge graphs. To model this, we describe a query as a decision function σ where

$$\sigma : \mathcal{P}_G \rightarrow \{accept, reject\}$$

Here σ takes as input a path (as defined in 3.3) and evaluates the path against some pre-defined rules. It then outputs a decision as to whether the path should be *accepted* or *rejected* based the specific function, i.e., query.

We can further define the space of all paths that a given σ function accepts or rejects. Let $X_\sigma \subset \mathcal{P}_G$ be defined as

$$X_\sigma = \{p \in \mathcal{P}_G | \ \sigma(p) \text{ is } accepted\}$$

X_σ is the set of paths that are **accepted** by the function σ.

This definition is extremely broad and enables path queries to be constructed. In fact, the stated definition of query above is undecidable. To show this, let's say we simply construct a function σ that takes any Turing machine M and input string x to that Turing machine. Let's say for any arbitrary path p, that $\sigma(p)_{M,x} = \{\ True$ iff M halts on $x\ \}$. In this case, to decide if we should accept p, we must first solve the halting problem. Our overly broad definition of pattern is difficult to use, so refining to a subset with more defined types of queries is desirable.

Here we seek to restrict our previously stated idea of a query. Now we limit our functions to only those whose domain is labels in the knowledge graph. These queries are also the forms that graph query languages such as Cypher take the form of.

Definition 3.7. Ontological query pattern. As defined in Definition 3.6, a query is a function σ that takes as input a path and decides whether the path should be *accepted* or *rejected*. Let us define a particular subset of these filter functions, $\hat{\sigma}$ where

$$\hat{\sigma} : \mathcal{O}_G \to \{true, false\}$$

That is, $\hat{\sigma}$ takes as input a collection of node and edge labels, and instance of a path (Definition 3.3) and returns a decision for if the path has ontological labels that are accepted. Let $X_{\hat{\sigma}} \subset \mathcal{P}_G$ be defined as

$$X_{\hat{\sigma}} = \{p \in \mathcal{P}_G |\ \hat{\sigma}(\delta(p, \phi, \psi)) \text{ is } true\}$$

That is, the set of all of the paths whose ontological reduction satisfy the pattern accepted by $\hat{\sigma}$.

In fact, we know this definition is substantially more computationally manageable. In work by Barceló, the author proved that resolving finding a single pathway through the graph that maps onto a set a specific set of node and edge labels is in NP (Barceló, 2013).

Finally, we seek to define the main theoretical contribution of this work. The idea here is the **semantic query pattern**; these are sets of node and edge labels that have meaning when seen

in a knowledge graph. This concept is inspired by the "semantic subgraph" from Hou et al. (Hou et al., 2022) and the "meta-path schemes" from Dong et al. (Dong et al., 2017).

Definition 3.8. Semantic Query Pattern. Semantic Query Patterns (SQPs) are query patterns that, when applied to a domain-specific knowledge graph, provide meaningful insights into the underlying relationships between ontological objects. Let σ be defined as

$$\sigma : \mathcal{O}_G \rightarrow \{accept, reject\}$$

Here, σ takes as input a collection of node and edge labels and returns a decision for if the labels are allowed. Let $X_\sigma \subset P$ be defined as

$$X_\sigma = \{p \in P | \sigma(\delta(p, \phi, \psi)) \text{ is } accepted\}$$

Less formally, X_σ is the the region of path space that is approved by σ.

Examples of these can be seen in Figure 3.7; these pathways are interesting combinations of node and edge labels that provide broad insight into the underlying principles that govern how ideas relate to each other. Such patterns have previously been constructed in the form of COPs and AOPs; our formalism seeks to generalize the concept into many fields.

Definition 3.9. Indicator Function The **indicator function** is a function which takes a boolean value as input and returns 1 if the boolean passed in is true, or 0 if the value passed in is false.

$$I(x) = \begin{cases} 1, & \text{if } x \\ 0, & \text{otherwise} \end{cases}$$

Definition 3.10. Rank. The **rank** value is a way to quantize the performance of a prediction metric into a integer value which places it's performance relative to how likely the tool is to predict other data more strongly. Let us have a reference data point x, set of data $Y = \{y_1, y_2, \ldots, y_n\}$, a target

data point $\hat{y}$ where $\hat{y} \in Y$, and some distance function d. To compute the **rank** value of x and $\hat{y}$, we compute $\forall y_i \in Y : d(x, y_j)$. We then order these predictions by their value. The rank is the where $d(x, \hat{y})$ in placed into this sorted list of predictions. Another way to write this definition of rank with the indicator function (Definition 3.9) as

$$rank(x, \hat{y}, Y) = \sum_{i=1}^{|Y|} I(d(x, y_i) \leq d(x, \hat{y}))$$

where we simply count all data points with a smaller distance value than our target.

The mean reciprocal rank (**MRR**) (Radev et al., 2002) has a history of long usage in computational research as a statistical measure of prediction quality. It provides us a singular value over our entire dataset, and can be used to compare the performance of our algorithm over multiple dataset sizes. MRR computes the reciprocal of each rank value for every pair. It sums each of these reciprocal ranks and then divides by the size of the dataset.

Definition 3.11. Mean Reciprocal Rank. The **MRR** is a value calculated on the performance of a set of predictions. For a set of data $Y = \{y_1, y_2, \ldots, y_n\}$, and specific data pairs $P = \{(x_1, y_1)$, (x_2, y_2), …, $(x_m, y_m)\}$, and set of trial we compute the rank of each of these pairs. We then take the reciprocal of this rank value, which provides us a value between 0 and 1 which reflects the performance of this rank.

$$MRR(P, Y) = \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{1}{rank(x_i, y_i, Y)}. \tag{3.4}$$

Definition 3.12. Hits@k The **Hits@k** function (Yin et al., 2017) is a measure which takes as input a set of pairs $P = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and set of data Y. We then compute how many of our predicted pairs have a $rank$ less than or equal to k. This value is used to show how likely we would be to capture the true answer if we only returned the top-k results. To calculate this, we need use the definition and notation of rank (Definition 3.10). We also need the indicator function (Definition 3.9).

$$Hits@k(P, Y, k) = \frac{1}{|P|} \sum_{i=1}^{|P|} I(rank(x_i, y_i, Y) \leq k). \tag{3.5}$$

Algorithm Description

In our work we assume we have a knowledge graph K =((V, E), T_V, T_E, ϕ, ψ). Assume we seek to find a family of semantic pathways over K connecting two specific node types: the starting node label **s** and the target node label **t**, where **s**, **t** $\in T_V$. In the COP example, s is a **Drug** and t is a **Disease**. In the movie pathway example provided in Figure 3.7, s is an **Actor** and t is a **Director**. We seek out a family of regular languages $\mathcal{L}$ that links node labels (s,t). We want these regular languages to express *high semanticity*. We seek to test the performance of $\mathcal{L}$ on n known pairs of specific nodes $\{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\} = P^+$. These are our positive training pairs. For any pair (a_j, b_j); $a_j, b_j \in V$ and $\phi(a_j) = \phi(b_j) =$ **s**, i.e., both a_j and b_j should be nodes from the knowledge graph K, and they should both have the same node label as the start label **s**. Additionally a_j and b_j should share some meaningful domain linkage. As an example, in our first case study, we look at drug-drug pairs (Section 3.3.5), so our start node label **s** = **drug**. Let's look at a sample from our training data (Section 3.3.5), `(simvastatin, lovastatin)`. Both of these nodes will map to the **drug** label, additionally, both of these drugs share a similar biological function, `hypolipidemic agent`, making them useful for finding an SQP that explains the mechanistic similarity of drug pairs. Additionally, we require a set of negative pairs P^-; these also must all have the node label **s**, but all negative pairs should share no meaningful domain linkage. Looking at an example of a negative pair from our case study, `(dexamethasone, levonorgestrel)`, these drugs operate with the mechansims `antenatal corticosteroids` and `endocrine and hormonal agent` respectively; these two mechanisms are vastly different and so these drugs are not meaningfully similar to each other.

Our Algorithm 4 works as follows. In line 2, we construct a regular language family $\mathcal{L}$ through combinatorially expanding the relationships between the nodes labels X_V and edge labels

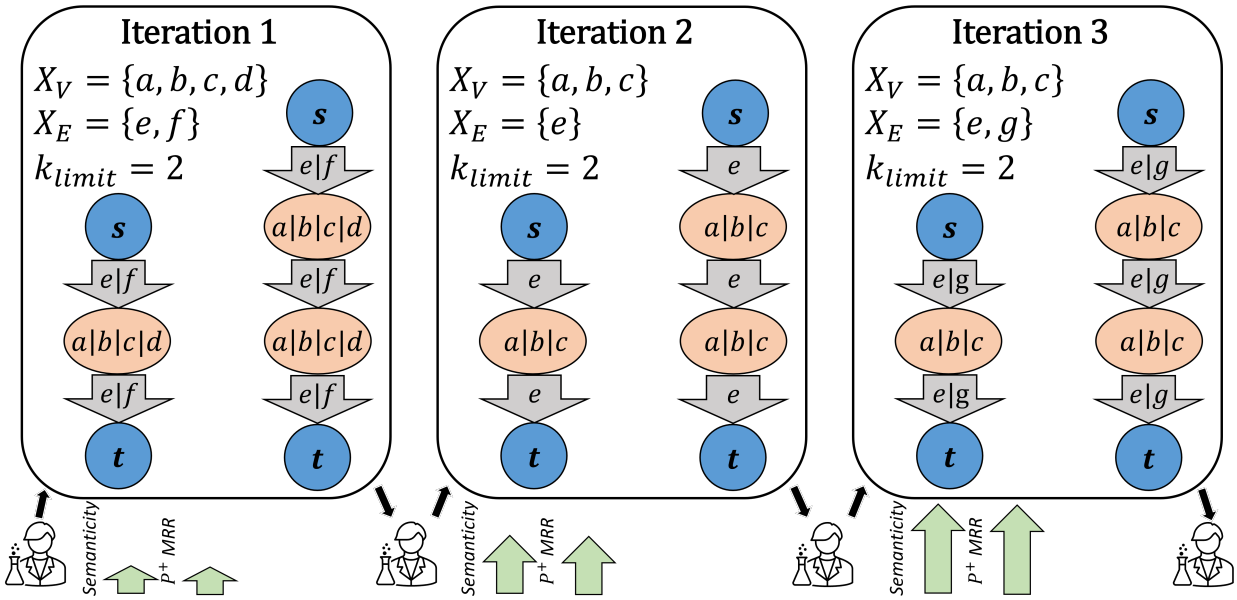


Figure 3.9: A visualization of three iterations of the *Ping-Pong* algorithm (Algorithm 5), with the expert passing in X_V ,X_E, k_{limit} in each iteration.

X_E (described in more detail below and in the function on lines 12-17). In line 3 we run the CompactWalks algorithm, described in Section 3.3.2, on the graph K, the embedding method $\mathcal{E}$, the constructed regular language $\mathcal{L}$, the positive pairs P^+, and the negative pairs P^-. When we run the CompactWalks algorithm, we receive a set of node embeddings for our positive and negative pairs; these node embeddings should reflect the relationship of these nodes on a graph filtered by our regular language. In line 4 we declare an empty list that will hold our constructed ranks. In the for loop on lines 5-7, we step through each positively linked pair we have provided in our input data. On line 6, we compute the rank (Definition 3.10) of the similarity of the nodes in the pair, compared to all other nodes in the input dataset. This rank value is then appended to our list on line 7. This same procedure is repeated on lines 8-10 for computing the relative ranks of the negative pairs. On line 11, the constructed regular language family $\mathcal{L}$ that connects **s** and **t** through node labels X_V and edge labels X_E, and the computed ranks of the positive pairs P^+ and negative pairs P^- are returned to the expert user to drive further evaluations and construction of SQPs.

The **RegexFamilyBuilder** function is detailed on lines 12-17 of Algorithm 4. Line 13 declares the family of languages to be an empty set. The for loop on lines 14-16 walks through each

Algorithm 4: SQP-Hunter(K, **s**, **t**, $\mathcal{E}$, X_V, X_E, P^+, P^-, k_{limit})

Data: Knowledge graph $K = ((V, E), T_V, T_E, \phi, \psi)$, starting node label **s**, target node label **t**, embedding algorithm $\mathcal{E}$, permitted node labels X_V, permitted edge labels X_E, positive pairs P^+, negative pairs P^-, expression length k_{limit}

Result: The constructed family of regular languages $\mathcal{L}$ that connect **s** and **t** through node labels X_V and edge labels X_E, and the ranks $ranks$ of scores of positive pairs P^+ and negative pairs P^- embedded via CompactWalks.

1 **begin**
2 $\quad\mathcal{L} \leftarrow$ `RegexFamilyBuilder(`***s, t,*** X_V, X_E, k_{limit}`); // See function on lines 12-17.`
3 $\quad\mathcal{V} \leftarrow$ `CompactWalks(`K,$\mathcal{E}$,$\mathcal{L}$,P^+,P^-`);`
4 $\quad ranks \leftarrow \emptyset$;
5 $\quad$**for** *each* $(n_1, n_2) \in P^+$ **do**
6 $\quad\quad r \leftarrow rank(n_1, n_2, P^+, \mathcal{V})$; `// Rank is the position of` n_1 `compared to` n_2 `relative to all other samples in` P^+`.`
7 $\quad\quad ranks.append(r)$;
8 $\quad$**for** *each* $(n_1, n_2) \in P^-$ **do**
9 $\quad\quad r \leftarrow rank(n_1, n_2, P^-, \mathcal{V})$; `// Rank is the position of` n_1 `compared to` n_2 `relative to all other samples in` P^-`.`
10 $\quad\quad ranks.append(r)$;
11 $\quad$**return** $\mathcal{L}, ranks$;
12 **Function** `RegexFamilyBuilder(`***s,t,***X_V,X_E,k_{limit}`)` **:**
13 $\quad\mathcal{L} \leftarrow \emptyset$;
14 $\quad$**for** $k \in \{1, 2, \ldots, k_{limit}\}$ **do**
15 $\quad\quad L_k \leftarrow \mathbf{s}(X_E X_V)^k (X_E)\mathbf{t}$;
16 $\quad\quad \mathcal{L} \leftarrow \mathcal{L} \bigcup L_k$;
17 $\quad$**return** $\mathcal{L}$;

possible path length value beginning at 1 and ending at the maximum expression length k_{limit} input to the algorithm. Line 15 constructs a family of regular languages L_k, which contains all regular language families with node labels X_V and edge labels X_E with k intermediate nodes separating the start node and end node. Line 16 adds the newly constructed language family L_k to the existing language collection $\mathcal{L}$. Line 17 returns the constructed regular language family.

Algorithm 5 works as follows. Line 2 initializes a boolean variable that indicates whether or not the while loop on lines 3-7 should run. We initialize it to $True$ to guarantee that this loop will run at least once. Line 4 seeks the following as input from the expert interacting with this

Algorithm 5: Ping-Pong Mining(K, **s**, **t**, $\mathcal{E}$)

Data: Knowledge graph $K = ((V, E), T_V, T_E, \phi, \psi)$, starting node label **s**, target node label **t**, embedding algorithm $\mathcal{E}$

Result: Family of regular languages $\mathcal{L}$ that connect **s** and **t** through node labels X_V and edge labels X_E.

```
1 begin
2     ContinueIteration ← True;
3     while ContinueIteration do
4         X_V, X_E, P+, P−, k_limit ← ExpertInput(); // Gather node/edge
              labels and positive/negative pairs from expert user.
5         L, ranks ← SQP-Hunter(K,s,t,E,X_V,X_E, ,P+,P−,k_limit);
6         PresentToExpert(L,ranks);
7         ContinueIteration ← ExpertInput(); // Allow the expert to
              choose if they're satisfied with the regular
              language and it's performance, or if they would like
              to continue.
8     return L;
```

algorithm: a set of node labels X_V, a set of edges labels X_E, a set of positive pairs P^+, a set of negative pairs P^-, and a pathway length k_{limit}. By asking the user for all of these, we enable them to experiment with how a meta-pathway they are constructing adapts to new data, or how data adapts to a new configuration of labels (this line is the **Ping** stage, where the expert passes their ideas to the algorithm). Line 5 calls Algorithm 4 on the inputs provided by the expert user. Line 6 presents the results of Algorithm 4 to the expert to use their intuition (this is the *Pong* stage, where the algorithm responds to the expert). Line 7 asks the user if they are satisfied with the current results, or if they would like another iteration of **ping-pong** with the algorithm. Line 8 returns the regular language to the user once they are satisfied with their SQP.

In Figure 3.9 we see an example of Algorithm 5. In each iteration the expert passes in X_V, X_E, P^+, P^-, and k_{limit}. The algorithm then builds a regular language family and runs CompactWalks with this family. The expert is then presented information on the MRR (Definition 3.11) performances of P^+ and P^-. In each iteration we see the regular language for $k = 1$ on the left and for $k = 2$ on the right. In *Iteration 1*, the expert submits node labels $X_V = \{a, b, c, d\}$ and edge labels $X_E = \{e, f\}$ as the first inputs to generate a regular language family . In *Iteration*

2, $X_V = \{a, b, c\}$ and $X_E = \{e\}$; the expert has removed node label d and edge label f from the possible solutions. In *Iteration 3*, $X_V = \{a, b, c\}$ and $X_E = \{e, g\}$, the expert has introduced edge label g as a potential candidate. At this stage the expert is satisfied with the resulting family of regular languages and concludes the trial.

3.3.5 Implementation and Case studies

We have created an implementation of the *Semantic Query* approach described in Section 3.3.4. We also seek to test our approach against two case studies to explore an expert user's ability to construct SQPs. Our implemnetation is described in Section 3.3.5. Our drug relatedness case study is described in Section 3.3.5. Our disease relatedness case study is described in Section 3.3.5.

Web Application

We implemented our ranking methodology in a web application. We chose to implement this software as a web app because it provides minimal friction to expert users who may want to use it without the complexity of setting up an appropriate Python environment or running a Docker container. As part of the Semantic Query Pattern project, we have provided an implementation of this web tool at `https://semantic-pathways.mml.unc.edu/`. This tool is a universal application for the construction of semantic query patterns against a limited set of data points. This tool is aimed at building with existing domain specific knowledge graphs and is for use by domain experts.

The web application was built using the Dash Python library [2]. A general outline of our methodology is as follows: (1) The expert generates a trial semantic pattern using the SQP-Hunter web app's interface. (2) The expert provides n positive pairs and m negative pairs. We implemented the tool using the Python Dash library. To dynamically render our figures we used a combination of Matplotlib and NetworkX. Whenever the user updates a selection of the *source*, *tail*, *meta-path length*, *node labels*, or *edge labels* we trigger an event which renders an updated figure. If no node

2. https://plotly.com/dash/

label or edge labels have been selected for a given location we mark it with a wildcard character (*). An example of this generated image can be seen in Figure 3.10.

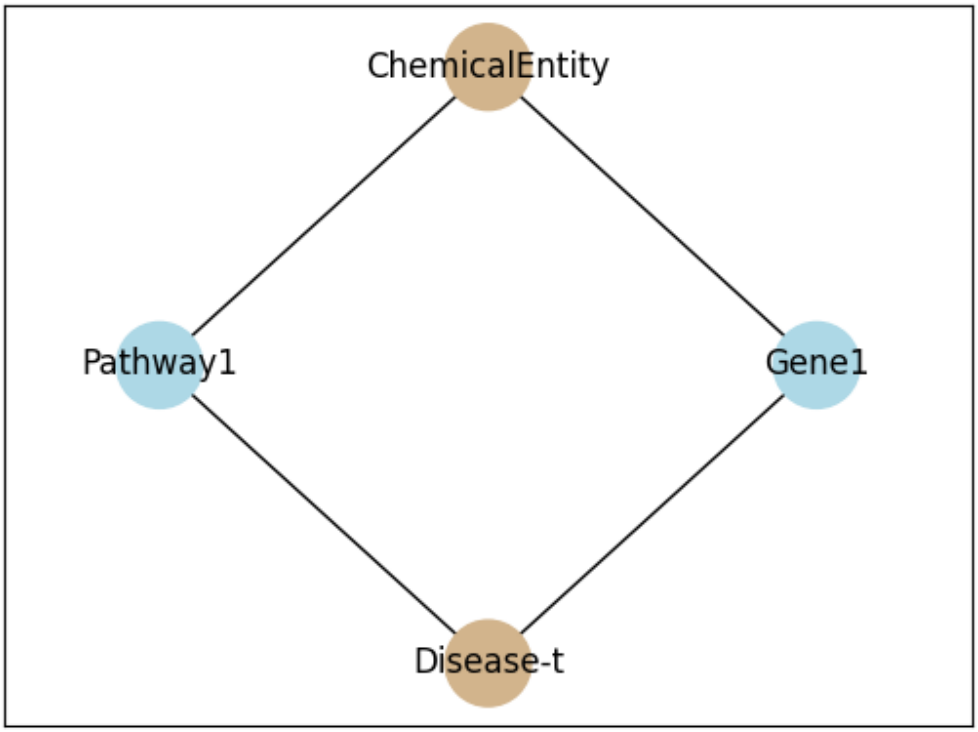


Figure 3.10: Example output of from the Semantic Query Pattern web application (hosted at `https://semantic-pathways.mml.unc.edu/`). This meta-path connects a `ChemicalEntity` to a `Disease` through either a `Gene` or `Pathway`.

The tool provides the user with a interface for providing *positive* pairs of ontological objects that are known to be similar through the expert's intuition, and *negative* pairs of ontological objects that share no similarity as far as the expert is concerned. These pairs should be of the same ontological class. An example positive pair is *betamethasone* and *dexamethasone*, which are two **drug** objects; in the biomedical space, these two drugs share a similar mechanism of action, i.e., their process of affecting the human body.

We have provided a live instance of our web application running on the Google Cloud Compute Engine. This was the instance used in our case study discussed below (Section 3.3.5). This web application was an allocated virtual machine (VM) running with the *e2-standard-2* configuration. The specifics of this configuration are as follows: 2 Intel Xeon(R) CPU@2.20GHz, 10 GB of persistent SSD storage, ephemeral IP addressing, and 8 GB of memory. The source code for the application can be found at `https://github.com/DnlRKorn/Semantic_Pathway_Generator`.

Experimental Objectives

The experimental goals of the Semantic Query Pattern project are as follows

1. To explore the intuition of experts as they develop semantic query patterns, by tracking an expert as they develop semantic query patterns in response to data.

2. Explore how these semantic queries developed by experts perform on validation data.

3. Explore the semantic meaning of expert produced queries.

Experimental Methodology

The CompactWalks methodology enables us to apply regular language families representing complex graph pathways onto a knowledge graph. We can then extract the subgraphs reduced by a regular language family from a specific knowledge graph using the CompactWalks methodology is discussed in 3.3.2 and can be viewed in Figure 3.8. Once these subgraphs have been extracted, a set of embedding vectors specific to the extracted subgraph can be created. We may then take similarity scores from various expert provided pairs of nodes. We have gathered pairs of nodes in the drug discovery domain.

Our experiments are as follows.

1. For the Drug Discovery problem domain, gather n pairs of *mechanistically* similar drugs, as determined by a biomedical expert. Ensure each of these drugs pairs operate under a

shared mechanism of action (i.e. betamethasone and dexamethasone are both antenatal corticosteroids and share a common mechanism of action). These initial ten pairs will be our **positive samples**.

2. Additionally, gather n pairs of non-similar drugs. We shall use these at **negative samples**.

3. To validate this claim, we shall run CompactWalks on our **test samples** using each of the t expert generated pathways. What we expect to see from this experiment is a monotonic increase in the MRR of the **positive test samples** and a monotonic decrease in the MRR of similarity of the **negative test samples**. That is, the final expert produced query should be very selective of our test data.

4. Have a biomedical expert review the queries and their semantic performance. Describe the semantic properties of these queries in the biomedical domain.

5. Analyze the final produced query in it's performance on separation of our test data. Additionally, test it's ability to *cluster* the data in hyper-dimensional space using the t-SNE algorithm (Van der Maaten and Hinton, 2008).

The expert will produce a series of increasingly semantic pathways using the tools Section 3.3.5. We expect each of these queries to of increased relevance to the task. When the expert is satisfied with the performance we will identify their final query as our *semantic query pattern*.

We run this last query on the validation data, to analyze the performance of the resulting semantic query pattern created by the expert's ability to separate data it was not trained on. We will also run this pathway in *t-SNE* (Van der Maaten and Hinton, 2008) to visualize how this final SQP embeds the nodes and see broad statistical patterns in how they are related.

Evaluation Metrics and Parameter Settings

To evaluate the performance of CompactWalks with each pair of data we use two measures. The *rank* (Definition 3.10) and *MRR* (Definition 3.11) (Radev et al., 2002). MRR takes the average

of the reciprocal ranks assigned to the true target drug for the pair. MRR is defined in Definition 3.11.

To compute the rank of how similarly we embed two vector, we must first fix our start vector. We then compute the cosine similarity (Eq 3.7) between our start vector against all other vectors in the our dataset. In our task, the dataset consists of every unique entity provided by our expert user in their submitted dataset. The rank is how similar our target vector is versus the similarity of all other embedded vectors. For example, if we have start vector $\mathbf{u}$ and target vector $\mathbf{v}$ has a rank of 4, this means that there are 3 embedding vectors in our dataset, $\mathbf{X}$, which has a higher cosine similarity to $\mathbf{u}$ than the value of $cosine(\mathbf{u}, \mathbf{v})$.

We compute the rank for each pair of nodes in our dataset, using the first node as the start and the second node as the target. We can then use this set of rank values for each pair to compute the MRR (Definition 3.11). We use MRR to get the average performance of the CompactWalks embeddings performance for our dataset of positive pairs and negative pairs.

As an additional metric of performance, we also compute the $Hits@k$ (Definition 3.12) for our dataset, which further describes the performance for specific benchmark values. We use Hits@1, Hits@3, and Hits@5.

To test the ability of our method to *cluster* pairs, we use normalized mutual information (NMI) (McDaid et al., 2011; Huang et al., 2014). The NMI is a well established measure of the ability of a methodology to cluster labeled points properly, evaluating the location of the embedding of each point, and comparing it to an ideal cluster. Opposed to traditional mutual information measures, NMI is bounded between 0 and 1, where larger numbers indicate better performance, which makes it easy to compare performance from different circumstances. The equation for NMI is as follows:

$$NMI(X, Y) = \frac{2 * I(X, Y)}{H(X) + H(Y)} \tag{3.6}$$

Where X is a set of predicted labels for data, Y is a set of true labels for the data, I is the measure of mutual information between the sets, and H is the measure of information entropy.

The distance function used in our experiments for comparing the embeddings of two nodes, which we use to compute the rank (Definition 3.10), is the cosine distance, stated as follows:

$$cosine(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} * \mathbf{y}}{|\mathbf{x}||\mathbf{y}|} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2} * \sqrt{\sum y_i^2}} \tag{3.7}$$

For our parameters, we choose to set the same defaults found as optimal in the CompactWalks paper. These are using the embedding algorithm $\mathcal{E}$ set to *Node2Vec*, p value=0.25, q value=0.25, walk-lengths of $l = 80$, and number of walks performed per node set to 5. We tasked our expert with producing semantic queries of lengths $k = 2$ and $k = 3$, enabling us to compare the differences in pathways and performance for different length patterns.

For the purpose of comparison, we seek to explore how our expert generated SQPs perform compared to an equivalent *non-semantic* pathway. When displaying our comparisons, generate the SQPs of length $k = 2$ and $k = 3$; we title the most discriminatory SQPs of our case study. We then feed these SQPs into CompactWalks and create semantic subgraphs (**SS-**), we title the experiments on these subgraphs **SS-k2** and **SS-k3** respectively. For comparison, we create non-semantic subgraphs (**NS-**) using meta-paths which will match pathways with any arbitrary paths of length $k = 2$ for the **NS-k2** case or any arbitrary paths of length $k = 3$ for the **NS-k3** case.

Drug Relatedness Case Study

We designed a case study to explore the capabilities of generating SQPs in the specific task of drug relatedness. The meta-paths generated here would seek out a way to elucidate relationships which explain how drugs of similar mechanisms interact. We tasked a biomedical expert and co-author of this study with selecting pairs of mechanistically related drugs (Section 3.3.5), and attempting to construct SQPs of length $k = 2$ and $k = 3$ using our webtool (Section 3.3.5) connected to the ROBOKOP KG (Bizon et al., 2019). The resulting SQPs are documented in Section 3.3.5. Additionally, utilizing the most discriminatory SQP, we performed clustering (detailed in Section 3.3.5).

Positive Pairs				Negative Pairs	
Drug A	Drug B	Mechanism	Indication	Drug A	Drug B
Fluoxetine	Paroxetine	Selective serotonin reuptake inhibitor (SSRI)	Neuropsychiatric agent	Fluoxetine	Lovastatin
Felodipine	Isradipine	Calcium channel L type blocker	Cardiovascular agent	Felodipine	Chlorpropamide
Nilotinib	Bosutinib	BCR-ABL inhibitor	Antineoplastic	Nilotinib	Brimonidine
Dexamethasone	Betamethasone	Corticosteroid	Anti-inflammatory	Dexamethasone	Levonorgestrel
Promethazine	Diphenhydramine	Histamine receptor H1 antagonist	Anti-allergic agent	Promethazine	Pantoprazole
Omeprazole	Pantoprazole	Proton pump inhibitor (PPI)	Gastrointestinal agent	Omeprazole	Diphenhydramine
Norethindrone	Levonorgestrel	Progesterone	Endocrine and hormonal agent	Norethindrone	Betamethasone
Apraclonidine	Brimonidine	Intraocular pressure lowering agent	Ophthalmic agent	Apraclonidine	Bosutinib
Glyburide	Chlorpropamide	Sulfonylurea	Antidiabetic agent	Glyburide	Isradipine
Simvastatin	Lovastatin	HMG-CoA reductase inhibitor	Hypolipidemic agent	Simvastatin	Paroxetine

Table 3.1: Our ten pairs of positive and negative drug relatedness data for training. For the positive pairs, we have included the mechanism by which these drugs interact with the body and their indication (their use in treatment of conditions).

Drug relatedness case study data The left hand side of Table 3.1 lists the positive pairs of related drugs which we used in the *ping-pong* stage of the algorithm with the biomedical domain-expert. These pairs of drugs share both pharmacological mechanism of action and indication (disease which a drug is prescribed by medical experts to treat). This means that these pairs should be well related and hopefully be able to uncover interesting regular languages which can be generalized over all the pairs. Additionally the righthand side of the Table 3.1 details the set of 10 negative pairs of drugs. These are drugs which do not share a mechanism and indication, which means they have no meaningful commonalities between them.

Positive Pairs				Negative Pairs	
Drug A	Drug B	Mechanism	Indication	Drug A	Drug B
Citalopram	Escitalopram	Selective serotonin reuptake inhibitor (SSRI)	Neuropsychiatric agent	Citalopram	Rosuvastatin
Nimodipine	Nisoldipine	Dihydropyridine calcium channel blocker	Cardiovascular agent	Nimodipine	Glimepiride
Imatinib	Dasatinib (Anhydrous)	BCR-ABL inhibitor	Antineoplastic	Imatinib	Carbachol
Prednisolone	Hydrocortisone	Corticosteroid	Anti-inflammatory	Prednisolone	Etonogestrel
Loratadine	Olanzapine	Histamine receptor H1 antagonist	Anti-allergic agent	Loratadine	Esomeprazole
Lansoprazole	Esomeprazole	Proton pump inhibitor (PPI)	Gastrointestinal agent	Lansoprazole	Olanzapine
Megestrol	Etonogestrel	Progesterone	Endocrine and hormonal agent	Megestrol	Hydrocortisone
Pilocarpine	Carbachol	Intraocular pressure lowering agent	Ophthalmic agent	Pilocarpine	Dasatinib (Anhydrous)
Glipizide	Glimepiride	Sulfonylurea	Antidiabetic agent	Glipizide	Nisoldipine
Atorvastatin	Rosuvastatin	HMG-CoA reductase inhibitor	Hypolipidemic agent	Atorvastatin	Escitalopram

Table 3.2: Our ten pairs of positive and negative data for testing. We have included the mechanism by which these drugs interact with the body and their indication (their use in treatment of conditions).

Similarly to the description above, in Table 3.2, we have an additional pairs of related drugs. We provide 10 pairs of positively related drugs (drugs sharing a mechanism and indication) and 10 negatively related drugs (those not sharing a mechanism and indication). We use these pairs to test and validate the performance of the expert produced SQPs and show we have produced a generalized pattern useful for explaining the relatedness of other drug pairs. Comparison between positively and negatively related pairs of drugs will demonstrate the power of generalized SQPs for discriminating between related and unrelated entity pairs. We detail the results of this in Figure 3.11.

Drug relatedness case study results Results of the Semantic Query Pattern experiments, presented in Figure 3.12, demonstrate the ability of the tool for assessing ”semanticity” of a given COP pattern for a particular application or task. In the case of these experiments, the task can be summarized as finding COP patterns which can help explain the mechanistic relationship between drugs sharing mechanism of action (MoA) and disease indication.

Figures 3.12a and 3.12b show the MRR for each COP query pattern illustrated in Figures 3.11b and 3.11a, respectively. Difference in MRR between positive and negative drug pairs gives an indication of how well the COP pattern is expected to differentiate additional meaningful pairs of drugs from non-meaningful or unrelated pairs. COP pattern 10 from from figure 3.11a demonstrated the greatest MRR difference among all other $k = 3$ query patterns, with MRR of 0.852 for the positive pairs and MRR of 0.227 for the negative pairs. This pattern is therefore expected to be the most ”semantic” pathway in the context of the current task. This pattern was also deemed highly semantic by a biomedical expert since it describes a meaningful sequence of events of increasing biological scale and complexity between a molecular initiating event and a clinical outcome. The least semantic $k = 3$ pathway, as ranked by the tool, was COP pattern 1 from Figure 3.11a. A biomedical expert reviewing this pathway agreed that this pattern was less semantic and arguably non-COP-like because it begins with ”ChemicalEntity” connecting to ”DiseaseOrPhenotypicFeature”. This step does not constitute a molecular initiating event, so this pattern could not be COP-like nor semantic.

Among the MRRs shown in Figure 3.11b for $k = 2$ query patterns, COP pattern 10 demonstrated the largest discriminatory power between positive and negative pairs with MRR of 0.666 for the positive pairs and MRR of 0.212 for the negative pairs. Interestingly, the second intermediate node in this pattern was left undefined such that any category of entity could fill this position. The 2nd and 3rd most semantic COP patterns (patterns 9 and 8 in figure 3.11b, respectively) were similar to pattern 10, except for the undefined second intermediate node was defined as “Pathway” (pattern 9) and “BiologicalProcessOrActivity” (pattern 8). A biomedical expert concluded that patterns 8, 9, and 10 were all semantic and COP-like for the same reasons

described for pattern 10 from Figure 3.11b. It is likely that pattern 10 was ranked most highly by the tool because it allowed for both "Pathway" and "BiologicalProcessOrActivity" entities to occupy the second intermediate node position, expanding the range of semantic and COP-like answers available for mechanistic explanation of drug pairs. Pattern 1 in Figure 3.11b was ranked least semantic by the tool and the biomedical expert again attributed this to the lack of molecular initiating event in the pattern. This pattern seems to describe a promising rule: if disease 1 and disease 2 share involvement of a particular gene, and a drug treats disease 1, then the same drug should treat disease 2. However, the tool demonstrates that this rule is actually unlikely to be helpful for the current task. This case example demonstrates the utility of the tool for pre-evaluating the suitability of rule-based inference for a given task.

The same experiments were repeated for test set positive and negative pairs to test ability to extrapolate results from the training set to the same task with different, but related, drug entities.

For $k = 2$ patterns, pattern 10 from figure 6 was also ranked most semantic (MRR of 0.642 for the positive pairs and MRR of 0.157 for the negative pairs) followed by pattern 8, then pattern 9. Since these three patterns are most highly ranked as semantic in both the training and test sets, this indicates the utility of these patterns for explaining new pairs.

For $k = 3$ patterns, pattern 10 from figure 5 was still ranked most semantic in terms of MRR difference (MRR of 0.626 for the positive pairs and MRR of 0.246 for the negative pairs). This result indicated that the same semantic COP pattern is generalizable to explaining relationships between new drug pairs sharing the same MoAs and indications as those from the training set pairs. It also indicates that explanations from this COP pattern are not sensitive to the identity of the specific drugs.

We performed *Hits@k* analysis of our most discriminatory patterns against non-semantic sampling of the graph using the CompactWalks methodology. We have labeled the most discriminatory path *SS-pattern*, which is the pattern we used to make our semantic subgraph. We generated ten unique embeddings by running CompactWalks ten times with a different random walk generated,

Method	Hits@		
	1	3	5
DeepWalk-NS-k3	0.12	0.21	0.28
DeepWalk-SS-k3	0.62	0.91	0.96
DeepWalk-NS-k2	0.14	0.25	0.3
DeepWalk-SS-k2	0.54	0.79	0.95
Node2Vec-NS-k3	0.09	0.15	0.28
Node2Vec-SS-k3	0.59	0.90	0.96
Node2Vec-NS-k2	0.16	0.25	0.33
Node2Vec-SS-k2	0.14	0.34	0.65

Table 3.3: The performance of our top performing patterns on the *Hits@k* metric (Definition 3.12) for our positive test pairs for the drug-relatedness case study. We compare the results on two different methods of random walk generation, DeepWalk and Node2Vec; and on semantic subgraphs (*SS-*) versus non-semantic subgraphs (*NS-*). To generate these results, we performed our embeddings with CompactWalks, ran our experiment ten times, and averaged the statistical performance.

using two different methods, *DeepWalk* and *Node2Vec*. Results of these experiments can be found in Table 3.3.

Drug relatedness case study clustering To perform our statistical validation experiments, we sought to evaluate the semantic performance of the highest performing patterns collected our biomedical expert for the drug relatedness case study. These patterns can be seen in Figure 3.11. We took the two semantic patterns are found from our approach in Section 3.3.5 on the ROBOKOP KG. The $k = 3$ *pattern* and $k = 2$ *pattern* are the patterns with the highest discriminatory power for length-3 and length-2 respectively. The $k = 3$ *pattern* is `ChemicalEntity` $\rightarrow$ `Gene` $\rightarrow$ `Pathway` $\rightarrow$ `Cell` $\rightarrow$ `DiseaseOrPhenotypicFeature`. The $k = 2$ *pattern* is `ChemicalEntity` $\rightarrow$ `Gene` $\rightarrow \star \rightarrow$ `DiseaseOrPhenotypicFeature`.

Table 3.4 shows the node-clustering results as measured by **normalized mutual information** (NMI) (Huang et al., 2014). Overall, the results for drugs indicate that embeddings with semantic subgraphs *(-SS)* using paths mined by our approach (i.e., $k = 3$ *pattern* and *the* $k = 2$ *pattern*) outperform embeddings with baseline *(-NS)* using the paths of arbitrary nodes/edges with the same number of intermediates.

Method	NMI	
	Drugs (Case 1)	Diseases (Case 2)
Deepwalk-NS-k3	0.578	0.583
Deepwalk-SS-k3	**0.805**	**0.727**
Deepwalk-NS-k2	0.618	0.634
Deepwalk-SS-k2	**0.766**	**0.608**
Node2Vec-NS-k3	0.580	0.580
Node2Vec-SS-k3	**0.803**	**0.733**
Node2Vec-NS-k22	0.622	0.636
Node2Vec-SS-k2	**0.764**	**0.608**

Table 3.4: Normalized mutual information data from the drug relatedness case study and disease relatedness case study. These values reflect the ability of a graph embedding method to cluster related data. Here we present the results from both the drug-relatedness and disease-relatedness case studies. We compare the performance of embeddings generated on the semantic subgraphs (*SS*-) and the non-semantic subgraphs (*NS*-).

To visualize the embedding vectors of drug groups, we used the t-SNE (Van der Maaten and Hinton, 2008) algorithm to make 2D projections of the embedding vectors with Deepwalk and CompactWalk framework using the semantic paths that mined by our approach, see Fig. 3.13. Each ground-truth drug cluster is shown in Fig. 3.13(b)–(c) in different shapes and colors, see Fig. 3.13(a) for the legend. Fig. 3.13(b)–(c) shows that using CompactWalks with semantic subgraphs that built from the $k = 3$ *drug relatedness pattern* and the $k = 2$ *drug relatedness pattern* makes the embeddings of similar drug nodes closer to each other, while making the embeddings of dissimilar clusters more distant. The clusters in Fig. 3.13(b)–(c) are also bounded using the convex-hulls approach (Barber et al., 1996), with 57.7% (15 in 26) for the $k = 3$ *pattern* and 61.5 % (16 in 26) for the $k = 2$ *pattern* of the drugs in the same ground-truth category placed in the same cluster.

We sought to quantify the ability of our pathways to *cluster* semantically similar nodes on a knowledge graph. We configured our CompactWalks to run for the following four settings: the $k = 3$ *drug relatedness pattern*, *non-semantic paths with 3 intermediates*, the $k = 2$ *drug relatedness pattern*, and *non-semantic paths with 2 intermediates*. We also varied the graph exploration method, using the DeepWalk and Node2Vec methods for generating random walks. We collected NMI values from 100 runs for each setting and for each of the two embedding methods, and then conducted

the two-sample t-test (Devore, 2015) for each pair of settings in ROBOKOP with each embedding method. We found that the NMIs for the $k = 3$ *drug relatedness pattern* and $k = 2$ *drug relatedness pattern* are significantly larger (p-value < 0.001) than the NMIs for each baseline (non-semantic paths with the same number of intermediate nodes). Based on these results, we posit that our approach is able to find meaningful semantic paths, such that constructing semantic subgraphs for embedding methods could lead to more accurate outcomes of clustering tasks.

Disease Relatedness Case Study

After validating the utility of the SQP tool for assessing semanticity of COP patterns, we sought to test its utility for a different task in the biomedical domain. For this task, we apply the SQP tool to assess semanticity of pathways that connect diseases or harmful phenotypes to the genes involved in their pathology. This task is particularly important for generating testable hypotheses for the use of a certain gene as a drug target. Generating strong drug target hypotheses could help accelerate drug target discovery, especially for rare disease with uncertain pathological mechanisms.

Disease relatedness case study data The left side of Table 3.5 describes 10 pairs of positively related disease pairs we used for training. The diseases in each pair share similar clinical symptoms and underlying biological mechanisms such that common genes are likely to play a role in the pathophysiology and/or treatment of both conditions. The goal is to identify generalizable semantic pathways to explain gene involvement in disease. Additionally the right side of the table provides have a set of 10 negative pairs of disease. These diseases do not share known symptoms or biological mechanisms that might connect them to a common gene with significant relevance to both diseases.

Disease relatedness case study results Similarly to the previous case study with COP patterns, our biomedical expert constructed 10 pathways with three intermediate nodes ($k = 3$ pathways) and 10 pathways with two intermediate nodes ($k = 2$ pathways) between DiseaseOrPhenotypicFeature and Gene nodes. These query patterns are shown in Figure 3.14. The $k = 3$ and $k = 2$ patterns are arranged from 1 to 10 in order of increasing separation of MRR between the positive and negative

pairs for each query. Figure 3.15 shows the MRR outcome of each test for $k = 3$ and $k = 2$ query patterns.

For the $k = 3$ patterns, the pattern with greatest discriminatory power between positive and negative pairs was pattern 10: `Disease` → `PhenotypicFeature` → `BiologicalProcess-OrPhenotypicFeature` → `PhenotypicFeature` → `Gene`. For this query pattern, positive pair MRR was 0.775 while negative pair MRR was 0.114. This pattern was also deemed to have high semantic meaning; the disease and gene entities are both associated with phenotypic features sharing a common biological process or activity. The two phenotypic feature nodes may be related or identical, but in either case we can infer that the gene node is involved in the biological process or activity related to both phenotypes, one or both of which is directly related to the disease.

For the $k = 2$ patterns, the pattern with greatest discriminatory power between positive and negative pairs was pattern 10: `Disease` → `ChemicalEntity` → `Gene` → `Gene`. For this query pattern, positive pair MRR was 0.483 while negative pair MRR was 0.131. This pattern was also deemed to have high semantic meaning; a chemical entity that is directly associated with the disease is also affecting a certain gene. It is therefore likely that the action of the chemical on the disease (either for good or for ill) is related to the chemical's action on that gene. That gene is functionally connected to another gene and we can infer that the two genes are involved in some common activity or pathway relevant to the disease. It is therefore possible that the second gene has a direct or indirect involvement in the disease.

The SQP Hunter tool is able to iteratively modify query patterns and trend toward greater discriminatory power between positive and negative disease pairs, just as it is able to do so for drug pairs. This demonstrates the wider applicability of the approach to numerous problems in the biomedical domain, provided the problem is clearly defined and appropriate positive and negative entity pairs are used to assess query pattern semanticity.

In the test case, we see that the pattern of increased differentiation mostly held true for the $k = 3$ case (Figure 3.15d). We see with the final pattern, the differentiation is 0.38 between the MRRs of the positive and negative samples. In the $k = 2$ case, our experiment had difficulty (Figure

3.15c), with the second pathway not being able to be found at all in our test pairs. Additionally, the performance of our experiments is less consistent on the test data pairs than on the training data pairs. We theorize that this inconsistency may result from the issue of disease-disease relationships being much harder to express, and ultimately being difficult to capture in meta-paths with only two intermediate nodes. Ultimately, the test pairs still showed significant differentiation, with 0.21 MRR on pattern 10.

We also performed *Hits@k* analysis of our most discriminatory patterns against non-semantic sampling of the graph using the CompactWalks methodology. We generated ten unique embeddings by running CompactWalks ten times with a different random walk generated. Results of these experiments can be found in Table 3.7.

Disease relatedness case study clustering Like we saw for drug-clustering in Section 3.3.5, Table 3.4 also shows the node-clustering results for diseases. The results for diseases indicate that embeddings with semantic subgraphs *(-SS)* using *path1* mined by our approach outperform embeddings with baseline *(-NS)* using the paths of arbitrary nodes/edges in same lengths. However, the *path2* did not have the same performance as *path1* did. Indeed, this finding mirrors results shown in Figure 3.15c; $k = 2$ SQPs lacked power to discriminate between positive and negative disease pairs compared to $k = 3$ SQPs.

Two semantic paths are found from our approach for ROBOKOP. For the disease relatedness case study, the $k = 3$ *disease relatedness pattern* is `Disease` $\rightarrow$ `PhenotypicFeature` $\rightarrow$ `BiologicalProcessOrActivity` $\rightarrow$ `PhenotypicFeature`$\rightarrow$ `Gene`. The $k = 2$ *disease relatedness pattern* is `Disease` $\rightarrow$ `ChemicalEntity` $\rightarrow$ `Gene` $\rightarrow$ `Gene`.

To visualize the embedding vectors of disease groups, we also used the t-SNE (Van der Maaten and Hinton, 2008) algorithm to make 2D projections of the embedding vectors found using the CompactWalk framework (with DeepWalk random explorations (Perozzi et al., 2014)) using the most discriminatory SQP, see Fig. 3.16. Each ground-truth drug cluster is shown in Fig. 3.16(b)–(c) in different shapes and colors, see Fig. 3.16(a) for the legend. Fig. 3.16(b)–(c) shows that using

CompactWalks with semantic subgraphs that built from $k = 3$ *pattern* makes the embeddings of similar drug nodes closer to each other, while making the embeddings of dissimilar clusters more distant.

The clusters in Fig. 3.16(b)–(c) are also bounded using the convex-hulls approach (Barber et al., 1996), with 66.7% (20 in 30) for *SS-k3* and 33.3 % (10 in 30) for *NS-k3* of the diseases in the same ground-truth category placed in the same cluster.

To see if the clustering results are significantly different across the two settings (i.e., **SS-** vs **NS-**), we collected NMIs from 100 runs for each setting and for each of the two embedding methods, and then conducted the two-sample t-test for each pair of settings in ROBOKOP with each embedding method. We found that the NMIs for the SS-path1 are significantly larger (p-value < 0.001) than the NMIs for the baseline (*NS-k3*). However, as we saw in Table 3.4, the NMIs are not significantly different between *SS-k2* and *NS-k2* for disease clustering, reflecting the poorer discriminatory power of k=2 patterns. Based on these results, again, we posit that our approach has the potential to find meaningful semantic paths, such that constructing semantic subgraphs for embedding methods could lead to a higher accuracy in downstream tasks.

3.3.6 Limitations

One limitation of the semantic query pattern approach is it's reliance upon expert users. Two experts may generate SQPs on the same training datasets; but may ultimately result in two drastically different meta-paths. This would reflect the differing biases of each expert, how they interpret the relationships of the data and which node and edge labels they believe may explain these connections. This limitation could also used for uncovering biases between the experts, and could be analyzed by the experts and potentially combining those SQPs from two expert may result in a stronger final pattern.

An additional limitation of our approach is that it is dependant on the data. This is a limitation also discussed of the underlying CompactWalks approach (Hou et al., 2022). The ability of our approach to develop pathways with discriminatory power is completely dependant upon a

the choice of positive and negative pairs by the expert user. Although in some cases, this may be a benefit of this method, as it places dependency completely on the data, leaving it completely flexible to learn from very diverse sets of input data.

We aid expert users that have a goal in mind. If these users are trying to ask the right questions, they may get the right answers. However, this presumes they have a problem they are exploring. If the expert is still at a stage of exploration in which they have no data to build upon, our tool cannot help them elucidate a pathway. Put another way; the SQP-Hunter reflects the intuition of the expert for *these* particular pairs. The higher quality the data and stronger the connections, the more meaningful the pathways mined will be.

3.3.7 Conclusions

We hope the production of SQPs for specific problem domains will primarily be used cases for experts. (1) The generated patterns can be utilized to explore the underlying mechanisms by which the provided data are linked, such as querying two datapoints in the knowledge graph and looking at the semantic subgraphs generated through the SQP; the overlapping areas for these highly similar compounds may unveil previously unknown information. (2) Using the generated SQP and novel data points as a method of hypothesis generation. Once a language $\mathcal{L}$ has been generated, an expert could run this query through the knowledge graph to identify candidates that may have a semantic linkage but be as of now unidentified.

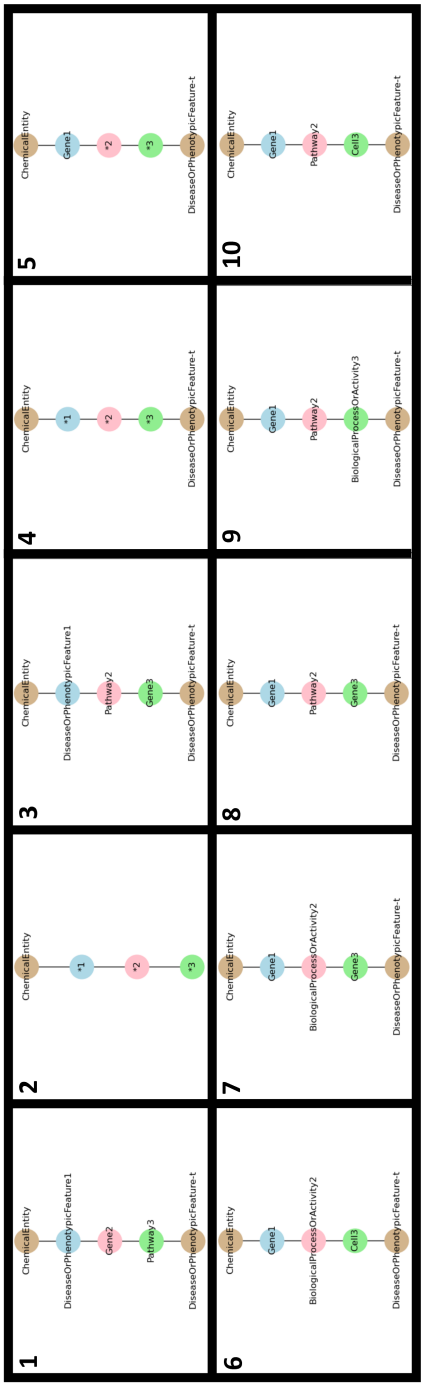
We hope that our *semantic query* approach and the associated implementation is of use to domain experts, who may seek to adapt it to their new problem domains.

3.3.8 Future Work

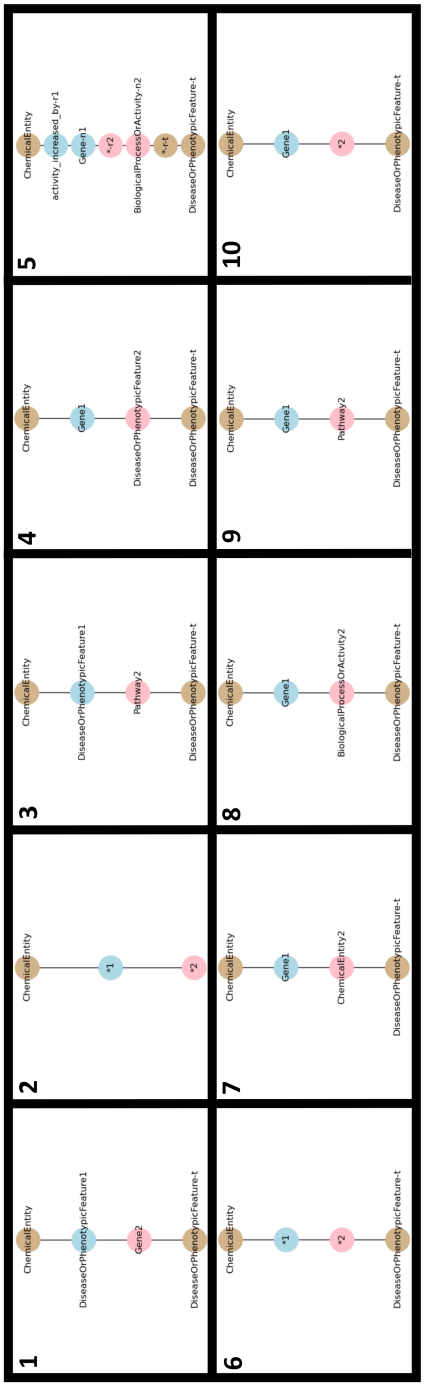
One interesting avenue of future progress on the SQP-Hunter algorithm would be complete automation away from the need for an expert user in the loop. One such approach could be an algorithm that takes inspiration from evolutionary methods (Bäck, 1996). This approach would attempt to dynamically evolve a complex query pattern just from a set of expert provided input pairs.

Because the space of potential graph patterns is so vast, this algorithm may require a long time to run or require a large amount of parallelizable compute power, such as a compute cluster.

A novel future approach would be more tooling for SQPs. We could develop a more complex web tool that provides expert users with custom ways to explore their generated pathways. Our current solution to this is to enable the experts to use Cypher to query their knowledge graph directly. But a unique tool could provide a more tight feedback loop, giving an expert who has mined novel semantic pathways the power to immediately search these pathways for connections between samples of interest. Additionally, this tool could include features to aid the user's exploration of individual answer subgraphs by providing an answer rank based on literature co-mentions between specific entities. This would provide expert users the ability to quickly find and explore studies related to pathways of interests. Such a tool could enable the creation of scenarios as we saw in the case study (Section 3.3.5) much more quickly.

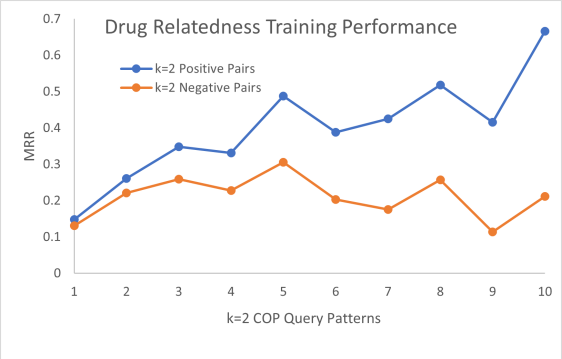


(a) Query patterns connecting *ChemicalEntity* to *DiseaseOrPhenotypicFeature* with two intermediate nodes ($k = 3$ pathways) chosen by the biomedical expert.

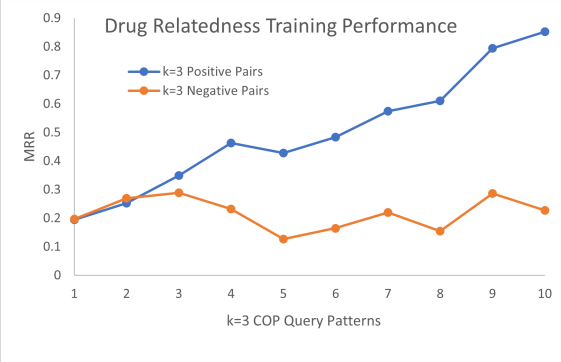


(b) Query patterns connecting *ChemicalEntity* to *DiseaseOrPhenotypicFeature* with two intermediate nodes ($k = 2$ pathways) chosen by the biomedical expert.

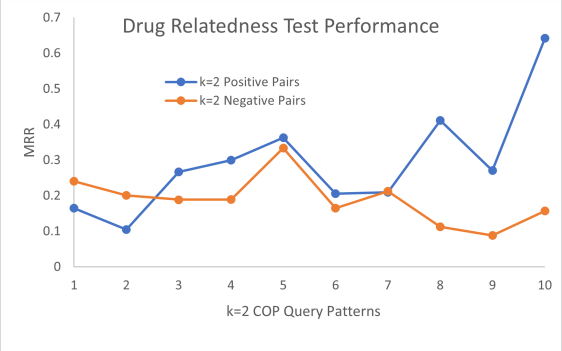
Figure 3.11: Query patterns generated by our biomedical expert for the drug relatedness case study. The patterns are arranged in ascending order from 1 to 10, with 10 being the pattern with greatest difference in MRR between positive and negative training pair sets and 1 being the pattern with the least difference.



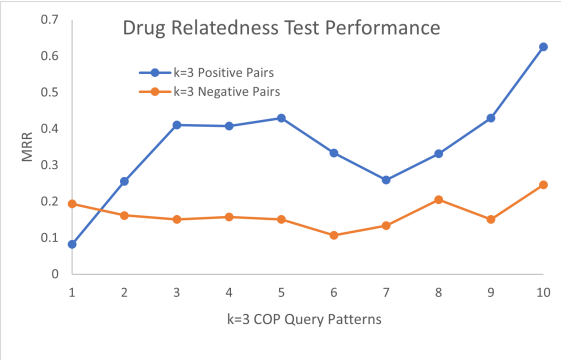
(a) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative training pairs on $k = 2$ COP patterns.



(b) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative training pairs on $k = 3$ COP patterns.

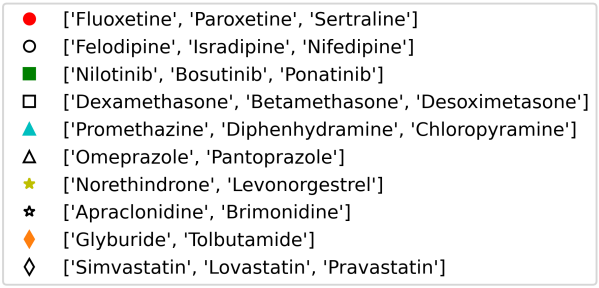


(c) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative test pairs on $k = 2$ COP patterns.

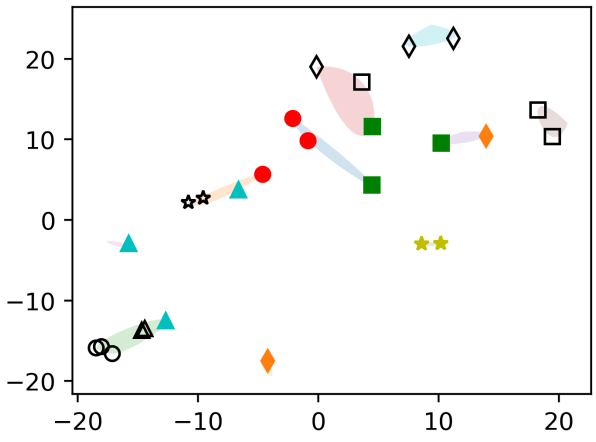


(d) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative test pairs on $k = 3$ COP patterns.

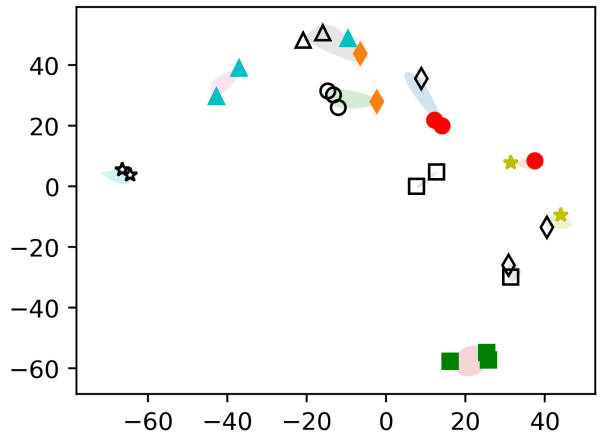
Figure 3.12: Mean reciprocal rank (MRR) output from drug relatedness SQPs for positive and negative pairs for $k = 2$ and $k = 3$ displayed in Figures 3.11b and 3.11a respectively. We compute this for our series of test and training data. Figures 3.12a/3.12b show the respectively performance on the training data in Table 3.1; Figures and 3.12c/3.12d show the performance of these pathways on the test data from Table 3.2.



(a) Ground-truth drug groups

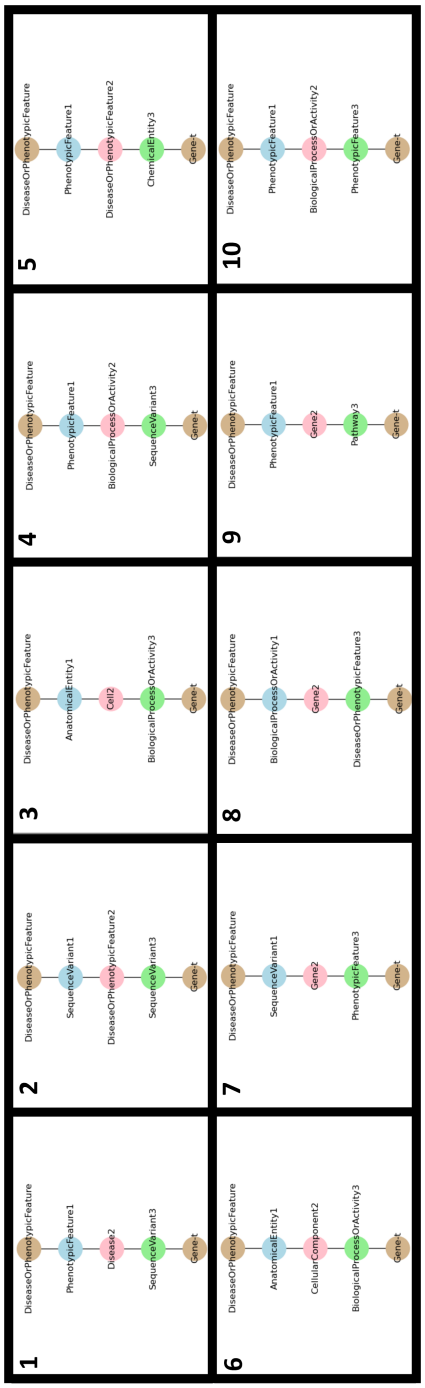


(b) Deepwalk-CompactWalks using the $k = 3$ *drug relatedness pattern*

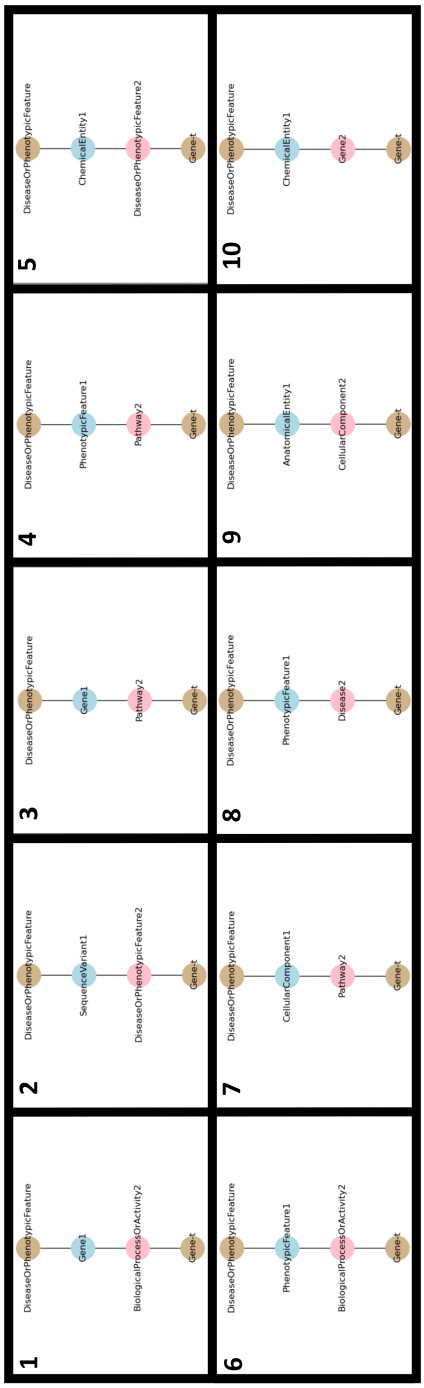


(c) Deepwalk-CompactWalks using the $k = 2$ *drug relatedness pattern*

Figure 3.13: 2D t-SNE (Van der Maaten and Hinton, 2008) projections of the embedding vectors generated by Deepwalk (Perozzi et al., 2014) on the ROBOKOP KG. In 3.13a we have a figure and all drugs. In 3.13b we see embeddings generated using the $k = 3$ *drug relatedness pattern*. In 3.13c we see embeddings generated with the $k = 3$ *pattern*. The clusters are visualized using convex hulls (Barber et al., 1996) with the predicted labels by k-means clustering (MacQueen et al., 1967).

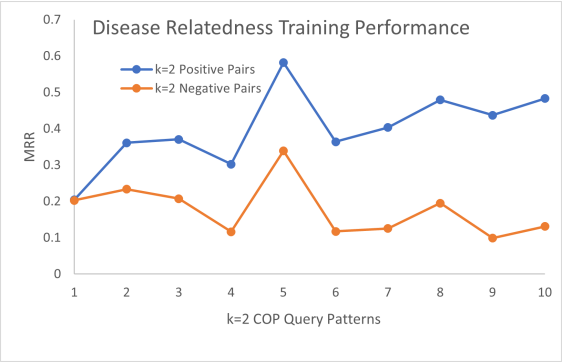


(a) Query patterns with three intermediate nodes (k=3 pathways) chosen by the biomedical expert.

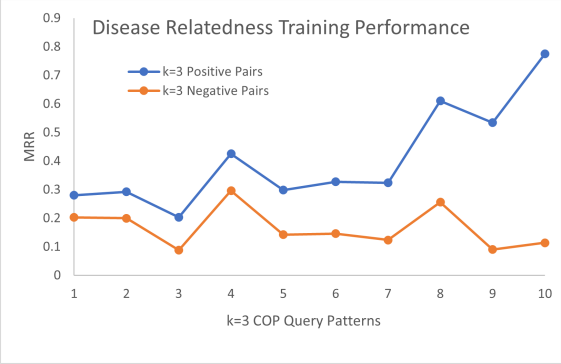


(b) Query patterns with two intermediate nodes (k=2 pathways) chosen by the biomedical expert.

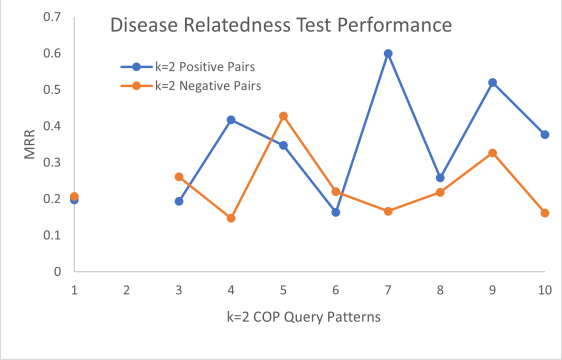
Figure 3.14: Query patterns generated by our domain-expert for the disease-relatedness case study. The patterns are arranged in ascending order 1 to 10, with 10 being the pattern with greatest difference in MRR between positive and negative training pair sets and 1 being the pattern with the least difference.



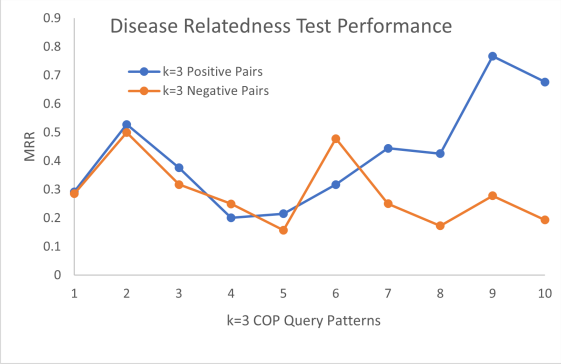
(a) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative test pairs on $k = 2$ COP patterns.



(b) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative test pairs on $k = 3$ COP patterns.



(c) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative test pairs on $k = 2$ COP patterns.



(d) Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative test pairs on $k = 3$ COP patterns.

Figure 3.15: Mean reciprocal rank (MRR) output from Semantic Query Pattern tool for positive and negative pairs for $k = 2$ and $k = 3$ displayed in Figures 3.14b and 3.14a respectively. We compute this for our series of test and training data. Figures 3.15a/3.15b show the respectively performance on the data in Table 3.5; Figures and 3.15c/3.15d show the performance of these pathways on the data from Table 3.6.

Positive Pairs			Negative Pairs	
Disease A	Disease B	Mechanism	Disease A	Disease B
Alzheimer's disease	Dementia	Mental deterioration	Alzheimer's disease	Hypertensive disorder
Type 2 diabetes mellitus	Type 1 diabetes mellitus	Insulin regulation	Type 2 diabetes mellitus	AIDS
HIV infectious disease	AIDS	Immune system dysfunction	HIV infectious disease	Non-alcoholic fatty liver disease
Heart disease	Hypertensive disorder	Cardiovascular system disruption	Heart disease	Skin cancer
Palsy	Cerebral palsy	Muscular disorder	Palsy	Chronic obstructive pulmonary disease
Melanoma	Skin cancer	Epidermis dysfunction	Melanoma	Type 1 diabetes mellitus
Synovitis	Rheumatoid arthritis	Inflammation	Synovitis	Headache
Asthma	Chronic obstructive pulmonary disease	Respiratory system	Asthma	Cerebral palsy
Fatty liver disease	Non-alcoholic fatty liver disease	Liver dysfunction	Fatty liver disease	Dementia
Migraine disorder	Headache	Cephalic pain	Migraine disorder	Rheumatoid arthritis

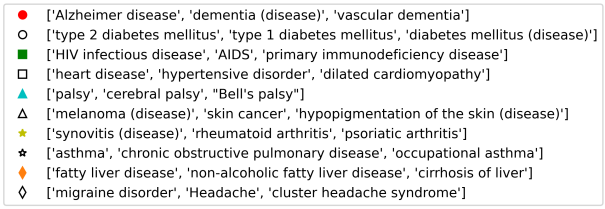
Table 3.5: Our ten pairs of positive and negative disease training data. Chosen by our expert for their similar clinical symptoms or biological mechanisms. We provide the mechanism which links the positive pairs.

Positive Pairs			Negative Pairs	
Disease A	Disease B	Mechanism	Disease A	Disease B
Irritable bowel syndrome	Crohn's disease	Gastrointestinal system	Irritable bowel syndrome	Anxiety disorder
Panic disorder	Anxiety disorder	Mental illness	Panic disorder	Intracranial sinus thrombosis
Pulmonary tuberculosis	Tuberculosis	Infectious bacterial disease	Pulmonary tuberculosis	Urticaria
Coronary thrombosis	Intracranial sinus thrombosis	Blood clotting	Coronary thrombosis	Brain cancer
Glaucoma	Cataracts	Visual decay	Glaucoma	Hemorrhoid
Herpes simplex virus	Genital herpes	Viral STDs	Herpes simplex virus	Tuberculosis
Atopic eczema	Urticaria	Dermal rashing	Atopic eczema	Genital herpes
Glioblastoma	Brain cancer	Tumors in brain	Glioblastoma	Crohn's disease
Anal fistula	Hemorrhoid	Rectal pain	Anal fistula	Syphilis
Gonorrhea	Syphilis	Bacterial STDs	Gonorrhea	Cataract

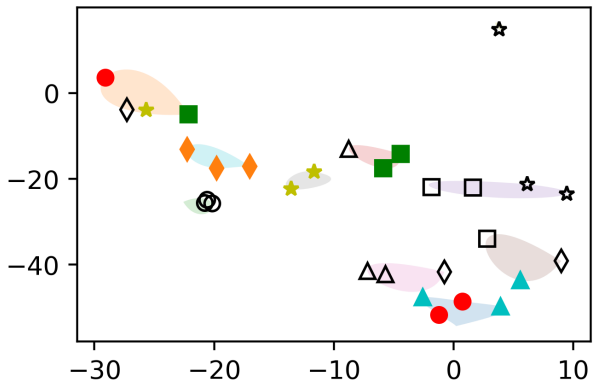
Table 3.6: Our ten pairs of positive and negative test data used to validate our disease relatedness pathways. Chosen by our expert for their similar clinical symptoms or biological mechanisms.

Method	Hits@		
	1	3	5
DeepWalk-NS-k3	0.24	0.35	0.46
DeepWalk-SS-k3	0.42	0.6	0.67
Node2Vec-NS-k3	0.27	0.38	0.5
Node2Vec-SS-k3	0.42	0.61	0.7
DeepWalk-NS-k2	0.27	0.6	0.63
DeepWalk-SS-k2	0.12	0.5	0.65
Node2Vec-NS-k2	0.28	0.57	0.68
Node2Vec-SS-k2	0.14	0.34	0.65

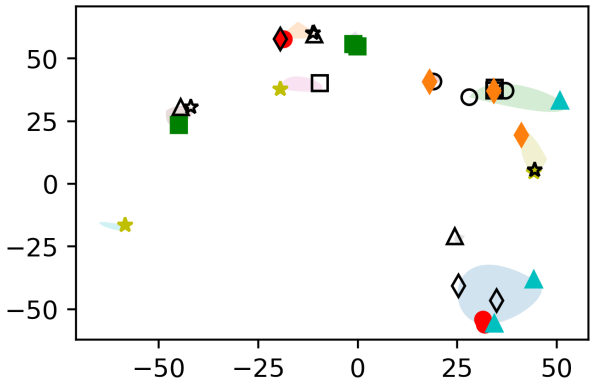
Table 3.7: The performance of our top performing patterns on the *Hits@k* metric (Definition 3.12 for our positive test pairs for the disease-relatedness case study. Performed the CompactWalks experiment ten times and averaged the statistical performance. We compare the performance of our embeddings generated on our semantic subgraphs (*SS*-) versus those on non-semantic subgraphs (*NS*-).



(a) Ground-truth disease groups



(b) Deepwalk-CompactWalks using $k = 3$ *disease relatedness pattern*



(c) Deepwalk using arbitrary paths of length-4 (NS-path1)

Figure 3.16: 2D t-SNE (Van der Maaten and Hinton, 2008) projections of the embedding vectors generated by Deepwalk (Perozzi et al., 2014) on the ROBOKOP KG for disease nodes shown in (a), obtained with CompactWalks framework using the $k = 3$ *pattern* in (b) and arbitrary paths of with three intermediate nodes in (c), respectively. The clusters are visualized using convex hulls (Barber et al., 1996) with the predicted labels by k-means clustering (MacQueen et al., 1967).

CHAPTER 4

Specialized Biomedical Knowledge Graphs

4.1 Introduction

The bioinformatics field poses a unique challenge from the computational perspective, which inherently seeks rigid order to all information. But the nature of medicine often directly relates to human expertise and intuition over quantifiable guidelines;(Woolley and Kostopoulou, 2013) this is baked directly into medical training in the form of residency requirements at hospitals, true expertise in the human body does not come from the classroom from real-life training. This gap between quantified knowledge and intuition is a particularly complex puzzle for computer scientists to tease apart. Yet, in the past, we have seen the opposite phenomenon occur, the example of Matthew Might, a computer scientist, utilizing computational tools to diagnosis his son, Bertrand Might, who was born with extremely rare and complex genetic defects.(Might, 2012) Dr. Might constructed a tool using the functional programming language Racket, and ultimately called it MediKanren. This tool was specifically designed to tease out connections in biomedical literature for his son's condition.(Might, 2012) These kinds of discoveries show the potential value of computational tools, enabling users without medical training to contribute to patient care. Bridging the gap between these two very different worlds could lead to an explosion of new medical discoveries. Doctors have notorious difficulties integrating computational tools into existing patient workflows.(Gawande, 2018) Although in recent years some medical schools have begun adding courses on artificial intelligence and computation to their curriculums,(Brouillette, 2019) these courses only cover the broad concepts and do not go into ideas such as programming and generating novel models. This

leaves a wide gap between computational capabilities and the ability for these capabilities to be leveraged in a meaningful manner.

We refer to the potential domain-expert users of our computational tools with expertise (but potential limited technical ability) as a **Subject Matter Expert** (SME). These SMEs take on many forms, such as doctors, nurses, geneticists, chemists, and biomedical researchers. We have identified a large gap in SMEs ability to engage with computational methodologies. In this aim, we seek to explore potential avenues of attack for this problem. Seeking to provide these users with more straightforward and comfortable methods to engage with our tools. These solutions will take different forms; software engineering projects enable large web applications with straightforward UIs and novel tools that can enable SMEs to discover the information they were not aware of.

4.2 Specialized Biomedical Knowledge Graphs

Reasoning Over Biomedical Objects linked in Knowledge Oriented Pathways (ROBOKOP, Section 1.5.1) is a major knowledge graph project produced by the Renaissance Computing Institute (RENCI) located at UNC-Chapel Hill. The ROBOKOP graph database aims to capture a broad and diverse set of biomedical data specializing in high curation and advances in interfaces that effectively allow SMEs to ask questions of the graph. Presently ROBOKOP integrates data from over twenty biomedical databases, capturing a variety of biomedical relationships, e.g., how chemicals interact with genes, phenotypes associated with diseases, anatomical entities' relationship to biological processes. Currently, the ROBOKOP KG contains 9.4 million distinct biomedical objects and 250 million edges linking these objects. Knowledge graphs provide incredible value on gathering information and generating hypotheses for broad information for the domain in which the graph was created. However, attempts to answer questions for extremely specific sub-domains are difficult. That is because knowledge graphs must choose levels of detail to represent any field. An overly detailed representation makes the information in the graph harder to understand and increases complexity for hypothesis generation. It is beneficial to lose some level of detail and capture a high-level understanding of a field. We propose that by existing broad biomedical knowledge graphs

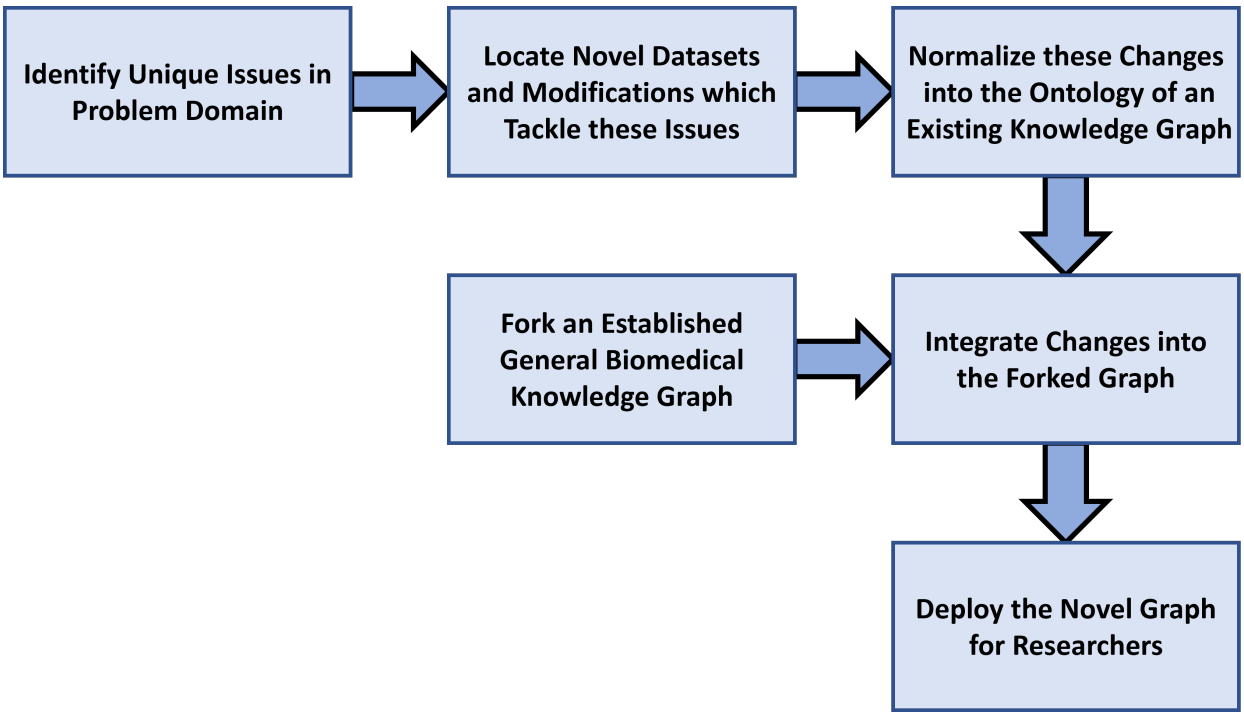


Figure 4.1: A high level overview of the workflow for creation of specialized knowledge graphs. The flexibility of this pipeline enables it to be utilized in multiple bioinformatics problems. It also enables the rapid deployment of specialized graphs in response to urgent circumstances, such as pandemics.

with information tightly related to a narrow problem domain, we can quickly and easily create specialized biomedical knowledge graphs which provide more robust and detailed answers inside a fixed well-defined biomedical area.

In this section, we explore the creation of knowledge graphs for these niche problem domains. We tackle this problem by choosing an existing knowledge graphs to act as the **primary KG**, a KG which serves as a primary repository of general knowledge. We then find specific biomedical databases which can be harmonized to the ontology which the primary KG already uses, and insert these data into our new **forked KG**. Depending on the desired use case, it may be advantageous to *down-weight* the edges which came from the primary KG and *up-weight* those edges which come from novel domain-specific datasets.

Figure 4.1 covers the workflow we have developed for *forking* KGs. We see an implementation of this workflow presented in the development of the COVID-KOP KG (Section 4.3) and the METAL-KOP KG (Section 4.4).

4.3 COVID-KOP

4.3.1 COVID-KOP - Introduction

With over 219 million cases and over 4.5 million deaths worldwide as of June 2021, and no Food and Drug Administration (FDA) approved treatments against this virus except for several drugs authorized for emergency use, there are continuing unprecedented global efforts to discover critical therapeutic treatments against COVID-19.(University and Medicine, 2021) These efforts already resulted in the identification and characterization of many SARS-CoV-2 proteins essential for virus replication,(Bobrowski et al., 2020) the pathogenesis of COVID-19,(Mousavizadeh and Ghasemi, 2021) and nomination of many drugs for clinical trials. Thousands of papers on COVID-19 and SARS-CoV-2 have appeared in the scientific literature since the beginning of the pandemic.(SciBite, 2021) There are many databases collecting data related to SARS-CoV-2;(of Health: Office of Data Science Strategy, 2021) however, the scientific literature concerning SARS-CoV-2 remains the largest repository of untapped biomedical data.(Bakken, 2020; Hunter, 2017) Knowledge graphs provide a key aid in the fight against COVID. Years have been spent integrating dozens of different databases into a singular source. Additionally, because the information comes from many different disciplines, it enables users who may not have specialized knowledge from any one field to leverage the discoveries and generate hypothesizes using those discoveries. In the COVID-KOP project, we sought to create a specialized "*fork*" of the ROBOKOP knowledge graph, which incorporated novel data relevant specifically to COVID-19 research. In doing so, we have shown that the creation of specialized knowledge graphs for specific problem instances can lead to positive results and increased ability to generate novel connections. Many of the edges contained in the COVID-KOP graph were taken directly from literature, which is also noisy and may pollute a more general biomedical knowledge graph.

4.3.2 COVID-KOP - Datasets

The Allen Institute for AI, in association with the White House, Georgetown University, the National Institute of Health (NIH) sought to release a large open-source corpus of biomedical medical literature in aid of the search treatments for the COVID-19 pandemic. This dataset was titled COVID-19 Open Research Dataset Challenge (CORD-19).(Lu Wang et al., 2020) Providing open-source academic papers is vital for natural language processing (NLP) work to be done on academic literature, which often involves various licenses and institutional permissions, and propriety APIs to access the full text of documents. Providing the documents in a more convenient format, such as flat text, and an ability to download papers in bulk was essential to our ability to create COVID-KOP. This dataset contained over 500,000 full text academic papers detailing work on COVID-19, in addition, existing work on epidemiology, virology, and historical pandemics have also been released into the dataset. To extract unique connections between biomedical entities from CORD-19; we leveraged Named Entity Recognition (NER) algorithms, which seek and identify biomedical entities in plain text. SciBiteAI is a private company based in the United Kingdom which primarily tackles issues of relationship identification, named entity recognition, and ontology development.(SciBite, 2021) In response to the COVID pandemic, our colleagues at SciBiteAI, a London based private company made their NER extractions from CORD-19 available for public use. By performing NER on the corpus of biomedical papers, and differentially weighting different sections of the paper (terms found in the abstract should be assumed to be of much greater importance than those in the body of the paper), we extracted relationships. These relationships were then merged with the existing ROBOKOP database. In silico proteomics information on COVID-19 related proteins was also incorporated into the COVID-KOP dataset. The Gene Ontology Annotations is a biomedical database focused on cataloging human and disease proteins, and their interactions in the body. (Ashburner et al., 2000; Gene Ontology Consortium, 2021) The data for the **GO** ontological identifiers for genes and proteins have been incorporated into the standard ROBOKOP database. The Gene Ontology Consortium provided and made publicly available information on the human proteins Sars-CoV-2 uses to enter the human cells, and predicted human proteins which the virus may target

after entry into human cells. This data is invaluable for enabling targeted drug discovery against the Sars-CoV-2 virus. This data is available at http://geneontology.org/covid-19.html. COVID-19 symptoms were also manually curated and gathered, as no comprehensive databases of symptoms were in existence when COVID-KOP was created. The symptoms were gathered from various online source:

1. https://www.cebm.net/covid-19/covid-19-signs-and-symptoms-tracker/,
2. https://covid.cd2h.org/N3C,
3. https://www.hematology.org/covid-19/covid-19-and-coagulopathy.

Additionally, one commentary was published containing a collection of COVID-19 symptoms which was also integrated.(Schett et al., 2020) This manually incorporation of data is a step that is not sustainable; as it would require human intervention each time the database were seeking to be updated. Human entry of information is also prone to error, especially in medical and clinical domains;(Goldberg et al., 2008) but it can be necessary when gathering novel data which has yet to be catalogued in any database.

4.3.3 COVID-KOP - Results

Figure 4.2 shows the resulting interface for the COVID-KOP web portal. Through quick efforts on both our team and the broader scientific community we were able to integrate the novel datasets, host, and deploy the COVID-KOP database and web portal in May 2020. This fast turnaround time was crucial towards the databases usability to the broader scientific community, and would not have been possible if not for the flexibility knowledge graphs provided us. Further studies done utilizing the COVID-KOP database by our collaborators yielded significant results.(Bobrowski et al., 2021) The goal of this study by Bobrowski et al was to utilize known mechanistic connections

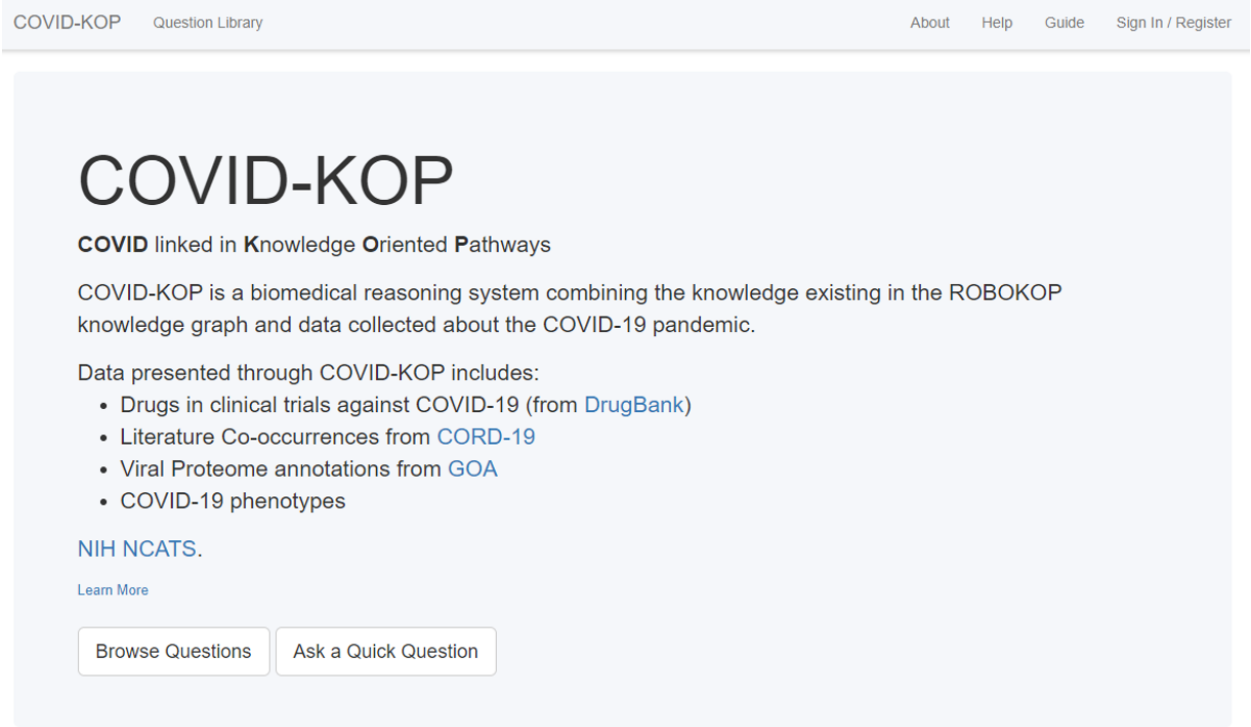


Figure 4.2: The COVID-KOP interface. The COVID-KOP tool enables users to query a fork of the ROBOKOP knowledge graph extended with COVID specific data. This web portal can be found at https://covidkop.renci.org.(Korn et al., 2020)

for drugs against the Sars-Cov-2 virus and then predict potential drug pairs which may, when taken in combination, produce a substantial effect against the virus.

COVID-KOP, used in combination general bioinformatics tools ROBOKOP (Section 1.5.1) and Chemotext (Capuzzi et al., 2018), enabled these researchers to quickly and efficiently determine what drugs had been tested against the virus. 76 drug candidates were selected. These tools were also used to determine the mechanisms by which these drug candidates operated. Pairs of drugs which operated by different mechanisms of action were combined and tested.

The study ultimately produced 73 combinations of 32 drugs. These drug combinations were tested in vitro against Sars-Cov-2 by the National Center for Advancing Translational Sciences (NCATS).

4.3.4 COVID-KOP – Conclusion

In response to the COVID-19 epidemic, we developed COVID-KOP, a knowledge base that integrates the existing ROBOKOP biomedical knowledge graph with information gathered from recently published biomedical information regarding COVID-19. The case study described here illustrates the utility of COVID-KOP in uncovering both known and unknown inferences between the drugs and COVID-19. COVID-KOP is freely accessible at `https://covidkop.renci.org/`.

4.4 METAL-KOP

4.4.1 METAL-KOP - Introduction

Metal implants have been in large scale use in humans for medical purposes since the late 1800s.(Hermawan et al., 2011) These implants have restored provided aid in healing, mobility, and quality of life to millions of patients with major injuries. Metal implants have made major improvements in multiple medical specializations, such as cardiovascular, with stents, orthopedics with artificial joints and bone plates, and dentistry where metal fillings are still necessary. Unfortunately, the body and metal are not intrinsically compatible. The body is a hostile environment to metal implants, causing them to wear and degrade over time. Metal implants are also incapable of biological functions, such as blood flow and bone conductance, making them treated by the body as an invading organism.(Goodman et al., 2014) This causes metal implants to become home to the phenomenon, metallosis, a poisoning of soft tissue in the body caused by a build up of metal debris.(Oliveira et al., 2015; Romesburg et al., 2010) Often this can lead to intense inflammation in the patient, elevated levels of metal in blood and urine samples, and in the most extreme cases local necrosis (the death of tissue surrounding the metal implant). Metallosis produces a great toll on both the physical and psychological state of patients, who already needing some form of implant, are already elderly or infirm.

4.4.2 METAL-KOP - Problem Statement

Another instance of specialized knowledge graphs is the METAL-KOP project. This project aims to create a fork of the ROBOKOP knowledge graph (Section 1.5.1), with modification specific to issues of metallomics. This project comes as an extension of work performed in collaboration with the Food and Drug Administration (FDA). The aims of the collaboration were to provide computational based approaches to identify metal compounds which may lead to metallosis. Additionally, we wanted to create justifications and potential pathways which may explain our hypothesizes. To help address these issues and create pathways specific for metallosis, our group sought to create METAL-KOP, a biomedical knowledge graph with information specific and relevant to the metallosis field.

4.4.3 METAL-KOP - Necessary Modifications

We must make modification to how the existing ROBOKOP architecture represents chemicals. That is because the standard ROBOKOP presents any atomic element as a singular node, for example, the chemical "Iron" is represented by one node *chemical_substance* with the SMILES string **Fe**. But when concerned with metals, the oxidation state of these elements is of note. We would want to replace the nodes representing metals with multiple nodes for each common oxidation state; Iron would then become $Iron^{II}$, $Iron^{III}$, $Iron^{IV}$, and $Iron^{VI}$. We also desire to put special properties associated with these oxidation states. Ions of the same element can have wildly different ionic radii and electronegativities, which can enable them to have very different effects when interacting with the human body. For example, the +2 state of cobalt, Co^{II}, is relatively safe and stable while the +3 state of cobalt, Co^{III}, is highly unstable and rapidly decomposes in an organic environment.(Jeffery and Hutchison, 1981)

Through our own experimentation, we have found that metals are not well defined by the MESH ontology (Lipscomb, 2000). This makes use of pre-existing tools, like Chemotext or standard ROBOKOP, more difficult. As an example, MeSH term D007501 (`https://id.nlm.nih.gov/mesh/D007501.html`) is used to represent the general concept of *iron*. In the

annotation of this MeSH term we see "*Fe(II) = FERROUS COMPOUNDS, Fe(III) = FERRIC COMPOUNDS*", where this term has been explicitly described to annotate different ionic states of iron. Therefore we must select a different ontological representation for chemicals as our standard representation in our Metal-KOP graph.

We have selected the Chemical Entities of Biological Interest (ChEBI) (Hastings et al., 2016) dictionary as our primary ontology of chemical in the METAL-KOP graph. The ChEBI dictonary aims to represent *small* chemical compounds accurately, which includes atomic metals. In the ChEBI ontology we can locate: CHEBI:18248 (`https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI:18248`) which represents atomic iron, CHEBI:29033 (`https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI:29033`), which represents $Iron^{II}$. Similarly we can locate CHEBI:29034 (`https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI:29034`) which represents $Iron^{III}$. This example shows the importance of ontology selection based upon the specific needs of a database, knowing we need higher resolution of metal compounds enforces that we choose a more granular representation of small molecules for the METAL-KOP knowledge graph.

4.4.4 METAL-KOP Datasets

To achieve the goals of creating a metal focused knowledge graph, we must also integrate biomedical databases particular specifically to metal ions. Metal PDB is a database which identifies how proteins in the human body can bind to different metal ions.(Andreini et al., 2013; Putignano et al., 2018) Additionally, we will include InterMetalDB which provides further information on metal binding.(Tran and Krężel, 2021) To get detailed lists of metal ions we can leverage the Comparative Toxicogenomics Database (CTD),(Davis et al., 2021; Mattingly et al., 2003) which provides enumerated connections from elements to their ionic forms.

The METAL-KOP is still currently in development. We plan to provide access to the METAL-KOP database to other scientific researchers and apply our own algorithms on this dataset in the future.

4.5 Future Work

4.5.1 TOXIC-KOP

Toxicology is the multidisciplinary study of toxic compounds, this field combines aspects of ecology, biology, chemistry, sociology, and many others (Timbrell, 2001). The field seeks to uncover the harmful substances around us, and develop actionable plans for the neutralization and removal of these substances. Large governmental organizations, like the Environmental Protection Agency (EPA) and the National Institute of Environmental Health Sciences (NIEHS) exist to ensure safe and effective standards around which compounds may exist within consumer products, medical devices, and introduce regulatory standards.

We aim with the TOXIC-KOP project to integrate existing datasets of toxic compounds and their negative effects on the human biosphere into a fork of the ROBOKOP knowledge graph. This idea already has incredible value in use of creation of Adverse Outcome Pathways (AOPs)(Society for Advancement of AOPs, 2021).

4.5.2 Rare Disease Knowledge Graphs

Rare diseases are conditions which affect fewer than 1 in 200,000 people as defined in the United States or 1 in 2,000 as defined in the European Union.(Valdez et al., 2016) Presently more than 600 therapies for rare diseases have been approved by the FDA.(Federal Drug Administration, 2018) However, as of 2010, over 7,000 rare diseases have been identified, and approximately 10% of the United States population suffer from at least one rare disease.(Valdez et al., 2016)

The drug discovery process is a lengthy one, consisting of many failures and a great deal of uncertainty. The drug discovery pipeline usually involves target identification and validation to detect molecules that may affect a disease state. In the discovery process, preclinical research trials include *in vitro* and *in vivo* efficacy, safety, and pharmacokinetic profiles, and clinical trials to establish safety and effectiveness in human subjects.(Hughes et al., 2011) Additionally, the path for finding drugs for rare diseases is even more strenuous.

Often, rare disease patients, with their support groups of family and friends must go through many year ordeals just to gain a diagnosis. Many doctors are not trained in the symptoms of rare diseases (with the large number and complex symptoms, such a process would be very difficult). Experts and specialists are sometimes simply non-existent for a condition, forcing patients or their support groups to become their own specialists and sometimes perform their own research. Within this process, they need to learn about the disease simultaneously with researchers and physicians.(Ekins and Perlstein, 2018)

One of the most famous cases is the story of Augusto and Michaela Odone, parents of Lorenzo Odone, who dedicated their lives to discover a treatment for their son's rare disease, adrenoleukodystrophy (ALD), and founded The Myelin Project, a research non-profit.(Odone and Odone, 1989) Augusto and Michaela never had any formal medical training, but they embarked on finding a cure for ALD. Adrenoleukodystrophy is a genetic disorder that results in the demyelination of neural fibers and degeneration of the adrenal gland, resulting in neurological instability and, ultimately, death.(National Center for Biotechnology Information (US), 1998; Odone and Odone, 1989) It was discovered that ALD causes the accumulation of saturated, long-chain fatty acids in the brain and adrenal cortex and leads to demyelination. Augusto and Michaela, with the help of researchers, eventually developed a treatment to break down these long-chain fatty acids by extracting acids from olive and rapeseed oils. This treatment was termed "Lorenzo's Oil.(Moser et al., 2005)" A study published in 2005 showed that, in certain cases, ALD could positively benefit from treatment with Lorenzo's Oil and prevent the progression of the disease.(Moser et al., 2005)

More recently, Matthew Might, a computer scientist and father of a child who has a rare disease involving an NGLY1 deficiency, became engaged in precision medicine and drug repurposing to find a treatment for his son's rare disease. He discovered that his son, as a result of the NGLY1 deficiency, also lacked N-acetylglucosamine, a vital amino sugar.(Chen et al., 2010b) Further research proposed that NGLY1 deficiency could potentially be treated with endo-beta-N-acetylglucosaminidase (ENGase) inhibitors.(Bi et al., 2017) A structure-based screening of a drug database and an electrophoretic mobility shift assay revealed that several drugs, most notably proton

pump inhibitors, could potentially be repurposed to treat NGLY1 deficiency.(Bi et al., 2017) This research revealed the possibility of drug repurposing for rare diseases and provided a direction for drug development and discovery for NGLY1 deficiency.

In response to the rare diseases, often computational tools provide more value than bench work. One strong use case for the specialized knowledge graph ontology would be automating the workflow discussed in Figure 4.1, and applying it to the thousands of rare diseases afflicting patients. We can then mine drug repurposing hypotheses (Section 1.4) to find unique drug-disease connections for each of these rare diseases.

CHAPTER 5

Conclusion

The work that embodies this dissertation set out to explore how biomedical knowledge graphs could be more productively leveraged to solve practical problems. We approached the issues of pathway ranking, pathway generation, and the creation of novel biomedical knowledge graphs. In Chapter 2 we looked at the issue of promiscuous nodes and created a novel ranking algorithm for knowledge graphs through that exploration. In Chapter 3 we sought to combine existing insights in the biochemical space, and use those insights to create specific pathways to glean information on how different biomedical ideas may be linked. And in Chapter 4 we looked at the issues of creating new knowledge graph, and presented case studies in different niche problem spaces; and a high level methodology for expanding this workflow to other domains.

The field still offers many more interesting questions and challenges that will need to be overcome in the upcoming years. Issues of accessing data, more complex and interesting ranking algorithms (hopefully built upon our work here). Federated knowledge graphs which can link dozens of existing biomedical KGs are on the horizon and may reshape the entire field.

In Chapter 2, we have detailed issues of hubs and promiscuous nodes in KGs. Through the formulation of promiscuity path scores we seek to leverage connections between nodes in KGs as a mechanism for measuring and predicting how these nodes relate. In our experimental results, we found that these promiscuity scores had meaningful statistical difference between positive and negative entity pairs in the knowledge graphs and their potential use in machine learning applications. We also extended these results to the case of federated knowledge graphs, showing the viability of promiscuity as a solid method for pruning results from these graphs, hopefully enabling them to return more meaningful paths to users quickly.

In Appendix 5.5, we have performed a detailed runtime and storage analysis on various algorithms designed to compute promiscuity scores for connected source and tail nodes. We have shown these values can be computed efficiently if the underlying promiscuity value connecting the two nodes is small. These results are summarized in Table A.1.

In Chapter 3, we studied the task of constructing task-specific meta-paths for knowledge graphs. In the biomedical context, we worked to develop *Clinical Outcome Pathways*, which catalog and understand how drugs and diseases interact with each other. Expanding on that, we introduced the *Semantic Query Pattern* idea, which constructed an expert driven approach at the construction of more domain/task-specific meta-paths from points of related data.

In Chapter 4, we studied issues of using generalized biomedical knowledge graphs to solve questions in niche subdomains. We sought out a general framework the creation of task specific biomedical knowledge graphs, and processes for identifying and integrating novel biomedical databases into these databases. We explored this fully in the *COVID-KOP* knowledge graph, developed specifically in response to the COVID-19 pandemic.

5.1 Importance of Biomedical Databases

One of the foundational pillars of the translational community is the ease and accessibility of data to mine. All knowledge graphs serve as meta sources, which aggregate existing information from primary sources (such as those detailed in Table 1.1). Each of these sources represent hundreds or thousands of hours of work from specialists in various domains, many of them performing the work on a voluntary basis. Keeping these databases high quality, free, and easily accessible is essential to the advancement of translational sciences.

5.2 Access to Medical Care

In the introduction we discussed how advances in medical science had provided humans with the resources and tools to treat those who had been afflicted with various conditions. Ailments which two hundred years ago would have had an extremely high mortality rate and essentially been

a death sentence to the patient can now be treated almost all western hospitals. Tuberculosis was a disease which at one point was a terrifying and widespread. Despite treatments of tuberculosis being known for over a century,(Murray et al., 2015) there was much difficulty getting treatment to patients.

Despite all our knowledge, in 2020, an estimated 10 million globally became infected with TB and an estimated 1.4 people million died from the disease. Tuberculosis still ranks as one of the top ten causes of infectious death globally. The majority of those infected were in Asia and Africa; with the majority of new cases emerging from just eight countries. The global disparity in our knowledge and our capability to treat a disease and it's continued presence should be of heavy concern to anyone working in medical and medical adjacent research.

Fortunately, we have seen a global decline in the infection rates of many diseases. There have been major victories for the medical community in recent times, such as the eradication of smallpox.(Fenner, 1982) We have shown the effectiveness of the global medical community, producing major multi-decade long health projects which have resulted in improved quality of life and an overall reduction in infection for the global population.

We have a duty as researchers and scientists to advance the goals of a healthy world. By democratizing the science of discovery and focusing on repurposing, not only of patented complex pharmaceuticals, but of all chemicals, whether they be made in a lab or found in tree bark, through efforts like drug repurposing (Section 1.4), we can take long existing pharmaceutical agents, many of which have readily accessible low-cost generic versions, and give these treatments further utility by finding all of their potential uses. This also serves as a way to escape the massive (and rising) costs of discovering novel pharmaceuticals, which must be passed onto the patients; or in the case of low patient rare diseases, is financially untenable.

5.3 Meta-research

The main concerns of this thesis have been in the applications of the work of others. The approaches discussed within our work is very high level, operating on knowledge graphs which serve

as the aggregation of collective thousands of hours of work from primary researchers, uncovering and labeling the specific biomedical entities, which we capture as nodes and further on, finding how these entities affect one another, which we capture as edges. We are in essence, performing research on how to further explore the research of others. It seems likely that as time moves forward and the ability to make meaningful discoveries in primary research become harder and more sporadic, more research man hours will be driven into *"meta-research"*, that is, intuiting connections from that which already has been discovered.

One core issue, that is apparent from the nature of this problem, is that meta-research will always be derivative, it is impossible to truly uncover new knowledge, all we discover are interesting hypotheses and potential ideas. We've argued here that many of these hypotheses have incredible merit which make them worthy of further exploration, but all output generated still requires verification. Until primary research is pursued which reinforces our hypotheses, the output from our methods don't provide

Ensuring the discoveries made are *actionable* is always key. Making sure that when these algorithms are being used to generate hypotheses, there is a road map for taking the potential research discoveries and verifying them.

This requires large scale collaboration, which is something we as computer science often have difficulty with. It is often easiest to view our datasets as provided from an oracle who we are complete separated from. This separation has been reinforced in a systemic fashion, where data is viewed as the only necessary aspect to computational research. Models may be built, trained, and tested all on entirely upon a sterile data, with no input from those who work in the fields to see if these models make sense, if there may be issues with the datasets.

5.4 Future Work

Details of future work specific to each of my aims are detailed in Sections 2.8, 3.3.8, and 4.5. We see great potential novel research projects which could be launched as an extension of this thesis. Projects which expand upon the mathematical ideas of promiscuity, expand upon

drug-disease connections even further, or present new tooling for domain-experts could be fruitful. Additionally, we see applying the ideas found in this thesis to other non-biomedical domains to be a promising avenue for future research.

5.5 Concluding Remarks

We see great potential in the use of biomedical knowledge graphs in both the theoretical case and practical practical applications. By gathering such diverse knowledge into one central location, we have potentially unlocked new avenues for research, and will accelerate the generation of new research hypothesis. Hopefully in the coming decades, knowledge graphs will see increased adoption across many disciplines, and will revolutionize how we approach asking questions and potentially all research questions as one of finding links in the web of ideas.

APPENDIX A

PROMISCUITY ANALYSIS

A.1 Promiscuity Analysis

A.1.1 Promiscuity Algorithm

In the earlier section, we looked at a version of psuedocode for the promiscuity algorithm which returns paths. In this section, we seek much more specific narrow goals of making claims of the runtime of a more focused version of the promiscuity algorithm.

We have developed an algorithm for the promiscuity analysis of graphs. This algorithm takes in two nodes, which we call the source and a tail; and looks for a pathway through the graph from the source to the tail. We utilize a naive, breadth first search (BFS) and depth first search (DFS), *for a fixed path length, seek the least promiscuous nodes which connect these nodes in the graph*. The node degrees of the least promiscuous pathway serve as a score for how strongly the two nodes are related. For example, if an extremely promiscuous pathway must be taken through a node like diabetes, we theorize that the linkage between the nodes may be spurious; unlike the case in which a pathway is discovered that has nodes of a lower degree.

Definition A.1. Knowledge Graph. A *knowledge graph (KG)* G is defined as a tuple $G = (V, E, T_V, T_E, \zeta, \xi)$, in which: (i) V is the set of nodes $v_1, v_2, ..., v_n$, with each node representing a data entity (e.g., a specific drug or disease); (ii) E is the set of (directed or undirected) edges $e_1, e_2, ..., e_m$, with each edge representing a relationship between two data entities represented by nodes in V (e.g., a drug `treats` a disease); (iii) T_V is the set of predefined node types (e.g., `drug` or `disease`); (iv) T_E is the set of predefined edge types (e.g., `treats`); (v) ζ is a node type mapping function $\zeta : V \rightarrow T_V$; and (vi) ξ is a link (edge) type mapping function $\xi : E \rightarrow T_E$. The *degree* of a node v in a KG G, denoted $degree(v)$, is the number of the edges that are adjacent to v in G.

Let us assume we are provided a knowledge graph according to the Definition A.1, $G = (V, E)$, a source node represented as $s \in V$, and a tail node be represented as $t \in V$. Define k as a hyperparameter which represents the length of the intermediate path. Let L be path of length k which connects s and t; so $L = \{v_1, v_2, \ldots, v_k\}$ where the nodes of L are in the graph, $v_1, v_2, \ldots, v_k \in V$, and pairs of nodes exist within the edges of G, $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k) \in E$, and $(s, v_1), (v_k, t) \in E$. This means that the graph can be traversed following the nodes in L starting at s and progressing over all nodes in L until the tail node t is reached.

Definition A.2. Path. We define a set of nodes $L = \{v_1, v_2, \ldots, v_k\}$ such that $v_1, v_2, \ldots, v_k \in V$ as a be path of length k which connects nodes s and t over Graph $G = (V, E)$ iff we have three properties.

1. There are k nodes in L, that is $|L| = k$.
2. The nodes of L are in the set of vertices for graph G, that is $\forall v_i \in L : v_i \in V$.
3. All pairs of sequential nodes in L must in the set of edges for G, that is $\forall v_{i-1}, v_i \in L : (v_{i-1}, v_i) \in E$.
4. The start node s must be connected to the first node v_1 in L, $(s, v_1) \in E$.
5. The final node in path L, v_k must have a connection to the tail node t, $(v_k, t) \in E$.

All of these rules taken in concert means that the graph can be traversed following the nodes in L, starting at s and progressing over all nodes in L until the tail node t is reached. To take the path from s over all nodes in L to t will take $k + 1$ "hops", or k intermediates.[1]

Definition A.3. Promiscuity Value. The promiscuity value for a path L on G is defined as

$$promiscuity(L) = max\{degree(v_1), degree(v_2), ..., degree(v_k)\}$$

1. This definition differs from our definition of *Path* provided in Chapter 2 (Definition 2.2). In that definition, we defined the length of the path by the number of edges, while here we define it by number of intermediate nodes. This is done for convince of notation in the following proofs. To convert between this definition to those of Definition 2.2 simply increase all path lengths by 1.

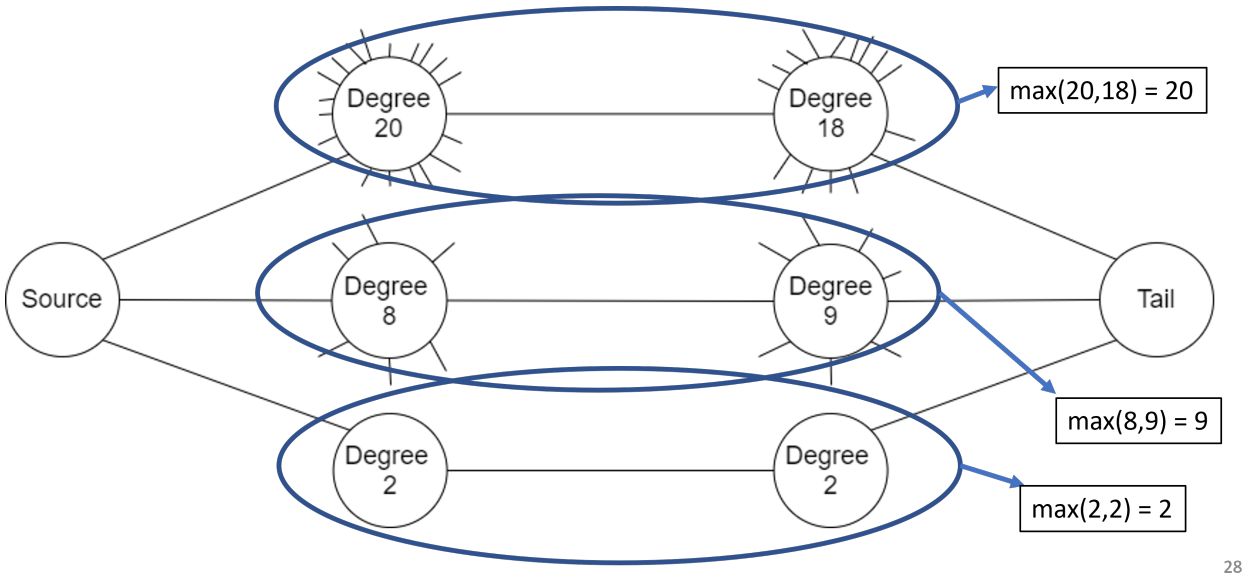


Figure A.1: An example of three pathways connecting a source and tail node. We calculate the *Promiscuity Value* (Definition A.3) for each of these pathways.

This metric measures the nodes of highest degree over an entire path.

Definition A.4. Promiscuity Score. The promiscuity score is a measure for any two nodes, which will be referred to as a source node s and tail node t, over a graph G for a provided path length k. This measure is global, that is, it is constant over the entire graph. We define the score as follows

$$\tau(s,t,k) = min\{\text{Promiscuity}(L) : \forall L \in G \text{ s.t. } s \xrightarrow{L} t\}$$

This metric measures the lowest achievable promiscuity score for all paths of length k connecting nodes s and t. If no path of length k exists which connects s and t we define $\tau(s,t,k) = \infty$

In the following sections we present three versions of an algorithm for computing the lowest promiscuity score $\tau(s,t,k)$ (Definition A.4). The promiscuity score is the maximum value of all degrees for each node along a pathway.

First in Section A.1.2, we explore a naive algorithm (Algorithm 6) which walks the graph in breath first manner, searching all paths between the source and tail. This algorithm keeps constant track of the highest degree it has seen on the current path it is walking. Whenever a path from the source node s to the tail node t is found, the promiscuity value (Definition A.3) of that path is checked against the current lowest promiscuity score we have seen in the execution of the algorithm, if the value is lower we update (line 17). This algorithm is a direct recreation of BFS with fixed length k. b is the branching factor of the graph G, which represents the maximal number of neighbors a node may have. The runtime of the naïve algorithm is $O(b^{k+1})$.

We explore an improvement to this algorithm in Section A.1.3. Our improvements to our ability to quickly identify this promiscuity this promiscuity score leverages the fact that we only need to analyze nodes with low degree to find minimum promiscuity. The second version of the promiscuity algorithm (Algorithm 7) leverages the fact that we only need to analyze nodes with low degree to find minimum promiscuity. If we sort nodes by degree before popping them off the queue for analysis, we can guarantee we have found the path with the smallest promiscuity score much earlier than the naïve algorithm. In fact, since the number of neighbors of a node is directly tied to its degree, the promiscuity score serves as an upper bound on the number of paths we must examine. The runtime of our improved algorithm (Algorithm 7) is $O(b * p^{k-1} * lg(b * p^{k-1}))$ (Theorem A.3).

Finally, we explore an alternative improvement on the naive algorithm in Section A.1.4. Our improvements here aims to find the promiscuity score using a small memory footprint. The third version of the promiscuity algorithm (Algorithm 8) . The runtime of our improved algorithm is $O(b^{k-1} * p * lg(p + (k-1)b))$ (Theorem A.4).

A.1.2 Naive First Search Promiscuity Score Algorithm

Here we describe a simple variation on a breadth first search graph search algorithm which finds paths of length k between nodes s and t. First we append all neighbors of the source node s onto the queue, in the loop on lines 3-6. For each path it explores, it simply keeps track of the largest node degree seen along said path, stored on the queue as $score_n$. When a path with a promiscuity

Algorithm 6: Naive algorithm for finding the promiscuity score $\tau(s, t, l)$.

Data: Graph G, source node s and target node t in G, path length l, number of paths k
Result: The promiscuity score $\tau(s, t, l)$ and the k least promiscuous paths of length l from s to t

```
1  begin
2      Q ← Queue(); // LIFO Queue
3      for n ∈ neighbors(s) do
4          depth_n ← 1;
5          score_n ← degree(n);
6          Q.append(score_n, depth_n, n);
7      τ(s, t, k) ← ∞;
8      while Q ≠ ∅ do
9          score_n, depth_n, n ← Q.pop(); // Gets next node on queue.
10         if depth_n < k then
11             for m ∈ neighbors(n) do
12                 depth_m ← depth_n + 1;
13                 score_m ← MAX{score_n, degree(m)};
14                 Q.append(score_m, depth_m, m);
15         if depth_n == k then
16             if n ∈ neighbors(t) then
17                 τ(s, t, k) ← MIN{τ(s, t, k), score_n};
18     return τ(s, t, l);
```

value lower than the lowest score seen so far is found, we update the current promiscuity score on line 17.

Lemma A.1. *Every path (Definition A.2) of length k will be explored by the by the while loop on lines 8-17 in Algorithm 6. By explored we mean that the promiscuity value will be calculated and line 17 will be executed.*

Proof. Proof by contradiction, assume there is some path (as defined in Definition A.2) $W = \{w_1, w_2, \ldots, w_k\}$ of length k which connects s and t. Assume this path is not explored by the while loop on lines 8-17. Let us consider the inductive case that each node in this path was not placed on the queue. Base case: Assume that node w_1 was not placed on the queue. Since W is a path, this means that $(s, w_1) \in E$. Since this is the case, w_1 must be placed on the queue with depth 1 in the loop on lines 3-6. Inductive case: Assume $i \in \{1, 2, \ldots, (k-1)\}$: w_i was placed on the

queue with depth i. Since there is no case in which the while loop on lines 8-17 may terminate before all nodes have been dequeued, at some point during the execution of the algorithm w_i must be dequeued. Since we have assumed W is path, by Definition A.2, $(w_i, w_{i+1}) \in E$, additionally we have assumed the depth of $i < k$, when w_i is dequeued, we shall pass the check on line 10 and run loop on lines 11-14. This loop must append w_{i+1} onto the queue with depth $i + 1$.

Hence, by induction, w_k must be placed onto the queue with depth k.

At some point during the execution of Algorithm 6, w_k must be dequeued with depth k. When this occurs the check on line 15 will be run. Since W is a path, by Definition A.2, $(w_k, t) \in E$, so we shall pass the check on line 16. Which will then cause line 17 to be executed for path W. Hence it cannot be the case that there is some path W of length k between s and t which is unexplored when Algorithm 6 terminates. □

Lemma A.2. *For any path (Definition A.2) $W = \{w_1, w_2, \ldots, w_k\}$ of length k connecting s and t, when w_k is dequeued on line 9, the $score_{w_k}$ value will reflect the promiscuity value (Definition A.3) of path W.*

Proof. Let $r = promiscuity(W)$. That is $r = max\{degree(w_1), degree(w_2), \ldots, degree(w_k)\}$. We seek to show when w_k is dequeued on line 9, $score_{w_k}$ will be equal to r. Let us perform induction over all of the elements path W.

Base case: w_1 is placed onto the queue with depth 1 and $score_{w_1} = max\{degree(w_1)\}$. This is because since W is a path, $(s, w_1) \in E$. So w_1 is appended to the queue in the for loop on lines 3-6. In this loop $score_{w_1}$ is calculated on line 5 as $degree(w_1)$.

Inductive case: Assume $i \in \{1, 2, \ldots, (k-1)\}$: w_i is dequeued from the queue with depth i and

$$score_{w_i} = max\{degree(w_1), degree(w_2), \ldots, degree(w_i)\}$$

We seek to show that w_{i+1} will be placed onto the queue with depth value $i + 1$ and

$$score_{w_{i+1}} = max\{degree(w_1), degree(w_2), \ldots, degree(w_i), degree(w_{i+1})\}$$

Since we have assumed W is path, by Definition A.2, $(w_i, w_{i+1}) \in E$, additionally we have assumed the depth of $i < k$, so when w_i is dequeued, we shall pass the check on line 10 and run loop on lines 11-14. Since w_{i+1} is a neighbor of w_i, we shall calculate $score_{w_{i+1}}$ on line 13. This calculation is $score_{w_{i+1}} = max\{score_{w_i}, degree(w_{i+1})\}$, which we can expand to be

$$score_{w_{i+1}} = max\{max\{degree(w_1), degree(w_2), \ldots, degree(w_i)\}, degree(w_{i+1})\}$$

Which is equivalent to

$$score_{w_{i+1}} = max\{degree(w_1), degree(w_2), \ldots, degree(w_i), degree(w_{i+1})\}$$

.

Hence, by inducting along all nodes in W, w_k will be placed onto the queue with

$$score_{w_k} = max\{degree(w_1), degree(w_2), \ldots, degree(w_k)\} = promiscuity(W)$$

□

Theorem A.1. *Algorithm 6 is complete, that is, for any nodes s and t and path length k, we shall return $\tau(s, t, k)$, the promiscuity score (Definition A.4). If no path can be found between s and t of length k, we shall return ∞.*

Proof. We know that by definition of promiscuity score (Definition A.4) there is some path $L = \{v_1, v_2, \ldots, v_k\}$ such that the promiscuity value (Definition A.3) $\tau(s, t, k) = promiscuity(L)$. Following the statement of Lemma A.1, we know all paths of length k must be explored by Algorithm 6, that is line 17 must run. Therefore L must be explored during the execution of Algorithm 6.

Following the statement of Lemma A.2, we know $score_{v_k}$ must have value $promiscuity(L)$. So when v_k is dequeued, line 17 will be executed to set the current promiscuity score tracked by the

algorithm to $score_{v_k} = \tau(s,t,k)$. Since no path may have promiscuity value lower than $\tau(s,t,k)$, the tracked value cannot be set any lower. So Algorithm 6 shall return $\tau(s,t,k)$ on line 18. □

A.1.3 Breadth First Search Promiscuity Value Algorithm

Algorithm 7: Breadth First Search Algorithm for finding the promiscuity score $\tau(s,t,l)$.

Data: Graph G, source node s and target node t in G, path length k
Result: The promiscuity score $\tau(s,t,k)$ from s to t over graph G

```
1  begin
2      Q ← PriorityQueue(); // min priority queue keyed on score_n
3      for n ∈ neighbors(s) do
4          depth_n ← 1;
5          score_n ← degree(n);
6          Q.append(degree(n) : score_n, depth_n, n);
7      τ(s,t,k) ← ∞;
8      t_neighbors ← BuildHashMap(neighbors(t));
9      while Q ≠ ∅ do
10         score_n, depth_n, n ← Q.extractMin();
           // Gets node on queue with lowest promiscuity score.
11         if degree(n) > τ(s,t,k) then
12             return τ(s,t,k) ;
13         if depth_n < k then
14             for m ∈ neighbors(n) do
15                 depth_m ← depth_n + 1;
16                 score_m ← MAX{score_n, degree(m)};
17                 Q.append(degree(m) : score_m, depth_m, m);
18         if depth_n == k then
19             if n ∈ t_neighbors then
20                 τ(s,t,k) ← MIN{τ(s,t,k), score_n};
21     return τ(s,t,k);
```

In this algorithm, we aim to improve on the naive approach (Algorithm 6). We perform three critical optimizations in our improved version of the algorithm. First, the queue on line 2 is a priority queue keyed on the degree of each node. The second is we build up a hashmap of the neighbors of the tail node t on line 8, which is then utilized in the check on line 19. Third, we introduce a new condition in the while loop on lines 9-20. This condition is on lines 11-12, and

it states that if the degree of the dequeued node is larger than the currently recorded promiscuity score $\tau(s, t, k)$, we should return. I claim that if we fall into this case, we have found the optimal promiscuity score, which we prove in Theorem A.2.

Theorem A.2. *The promiscuity algorithm is guaranteed to return the optimal promiscuity score $\tau(s, t, k)$ (Definition A.4) over $G = (V, E)$ for any nodes $s, t \in V$ and any value k. If no path exists between s and t, we shall return ∞.*

Proof. Let $G = (V, E)$ be a graph, $s \in V$ be a source node, $t \in G$ be a tail node, and k be some non-negative integer. Let us call the value returned by Algorithm 7 when run on s, t, t, and G returns value r. Let p be the true minimal promiscuity value for s and t over G with path length k. If it is the case that no path can be found between s and t, we say that $p = \infty$ (as per Definition A.4).

Let it be the case that $r \neq p$, that is the returned value of Algorithm 7 is inconsistent with our expected result. There must be one of two cases: **Case 1:** There does not exist a path of length k which connects s and t, thus the minimal promiscuity value $p = \infty$. **Case 2:** There does exist a path of length k between s and t.

Case 1: There does not exist a path of length k which connects s and t. In this case the minimal promiscuity value $p = \infty$. If the value of p were not infinity, it implies there exists a path connecting s and t (by definition of Promiscuity in Definition A.4. Since $r \neq p$, r must be some value which is not infinity. Therefore line 20 must run at least one time during the runtime of the algorithm, as that is the only place in the entire algorithm for which the return value $\tau(s, t, k)$ is updated. Let us call the node for which line 20 runs v_k. For any dequeued node, line 20 can only run if v_k has both (1) a depth value of k (line 18) and (2) t is a neighbor of v_k (line 19). However, if that is the case, then v_k must have been put on the queue

exists on some path of length k which connects s and t, which we have assumed does not exist. Hence, it cannot be the case that r is any value other than infinity if no path of length k connects s and t. □

Case 2) There does exist a path of length k which connects s and t. Since there is at least one path which connects s and t, it cannot be the case that $p = \infty$. By definition of promiscuity there must exist some path $L = \{v_1, v_2, \ldots, v_k\}$ which connects s and t such that promiscuity(L)=p. We know $r \neq p$ so we have either $r < p$ or $r > p$. We shall refer to these as Case 2a and Case 2b respectively.

Case 2a) Assume the promiscuity algorithm returns some value r such that $r < p$. It must be the case that for some node w_k, line 20 of the algorithm ran and set the value of $\tau(s, t, k)$ to r. Since line 20 ran for node w_k, it must be the case that the depth value of node w_k was k (or else the check on line 18 would fail). It must be the case that both $score_{w_k} = r$. Since w_k was placed on the queue with $score_{w_k} = r$, it must be the case that another node enqueued it; which we shall call w_{k-1}. This node must have had a depth of $k - 1$ and it must be the case that $(w_{k-1}, w_k) \in E$. When line 16. Therefore, we know that $\tau(s, t, k)$, as defined on line 20, was set to r, which implies that both $degree(w_k) \leq r$ and that the $score_{w_k}$ of w_{k-1} was less than or equal to r.

Inductive Step: Assume that node w_i places w_{i+1} on the queue with a $score_{w_{i+1}}$ less than or equal to r, and with a depth value $i + 1$. Let us define the node which enqueued w_i as w_{i-1}. It must be the case that there exists an edge $(w_{i-1}, w_i) \in E$. It must also be the case that 1) the $score_{w_{i-1}} \leq score_{w_i} \leq r$ and that 2) degree$(w_i) \leq r$. If either of these were not the case, then $score_{w_i} > r$.

By induction, we know there must be a whole chain of nodes $W = \{w_1, w_2, w_3, \ldots, w_{k-1}, w_k\}$ of increasing depth value where

$$\begin{aligned} \forall i \in [1, k]\text{: degree}(w_i) \leq r \\ \forall i \in [1, k-1]\text{: } (w_i, w_{i+1}) \in E \end{aligned} \tag{A.1}$$

Additionally, we know that the only way a node of depth 1 may be placed on the queue is in lines 3-6, which may only enqueue neighbors of the source node. Hence, $(s, w_1) \in E$. Therefore, W is a valid path of length k connecting the source and tail. And Promiscuity$(W) = max(\text{degree}(w_1), \text{degree}(w_2), \ldots, \text{degree}(w_k)) \leq r$. But promiscuity$(W) \leq r < p$, which violates

our assumption that p was the minimal promiscuity value for all paths of length k connecting s and t. Hence, we cannot return a value r which is less than p. □

Case 2b) Assume the promiscuity algorithm returns some value r such that $r > p$. Let a path with promiscuity value p be represented by $L = \{v_1, v_2, \ldots, v_k\}$. Additionally, let the path which set best_score to r be represented by $W = \{w_1, w_2, \ldots, w_k\}$. It must be the case that either 1) v_k is dequeued before w_k or 2) v_k is dequeued after w_k. If 1) then v_k must set the $\tau(s, t, k)$ to a value less than or equal to p, and line 20 will not update the value of τ to r, since $r > p$. So, r would not be returned, and it must be the case that 2) v_k is dequeued after w_k. We know that Promiscuity(W) = max(degree(w_1), degree(w_2), ..., degree(w_k)) = r. Hence, there must be at least one node in path W with degree$(w_i) = r$. But the queue defined on line 2 of the algorithm is a priority queue keyed on the degrees of nodes, and we know that, by definition of promiscuity,

$$\forall j \in \{1, k\} : \text{degree}(v_j) \leq p < r = \text{degree}(w_i). \tag{A.2}$$

Hence, all nodes in path L must be enqueued and dequeued before w_i is dequeued. Since w_i precedes w_k on path W, w_i must be dequeued before w_k. So, it cannot be the case that v_k is dequeued after w_k.

Therefore, the algorithm may not return value $r > p$.

Since for inputs s and t and $\tau(s, t, k) = p$ Algorithm 7 may not return $r < p$ by the proof shown in Case 2a) and Algorithm 7 may not return $r > p$ by the proof shown in Case 2b); it must be the case that Algorithm 7 returns $r = p$.□

□

Theorem A.3. *The Breadth First Search Variant of the promiscuity algorithm (Algorithm 7) is guaranteed to have a worst case runtime of* $O(b * p^{k-1} * lg(b * p^{k-1}))$.

To show Theorem A.3 we must first prove two lemmas.

Lemma A.3. *No node with degree larger than p, where p is the promiscuity value for source node s and tail node t over graph $G = (V, E)$ with path length k, may be dequeued during execution of Algorithm 7 and not cause the algorithm to terminate on line 12.*

Proof. Here we show that any node we dequeue and run line 12 for must have degree at most p, where p is the promiscuity value for source s and tail t over graph G with path length k. By definition of promiscuity score (Definition A.4) there must exist some path $L = \{v_1, v_2, \ldots, v_k\}$ which connects s and t such that promiscuity$(L) = p$. Let us assume at any point during the execution of the while loop on lines 9-20 we dequeue a node z, where degree$(z) > p$, but we do not return on line 12. Since two nodes may not be dequeued at once, it must be the case that either Case 1) z was dequeued before v_k was dequeued or Case 2) z was dequeued after v_k was dequeued.

Assume Case 1 is true, that is z was dequeued before v_k was dequeued. Either a) v_k is still in the queue or b) v_k was never enqueued. We know that since v_k is part of path L; that degree$(v_k) \leq p$; if this were not the case, the promiscuity score of path L could not be p (by Definition A.3). If v_k is in the queue, then it must be dequeued before z since we have a priority queue keyed on degrees of nodes, and degree$(v_k) \leq p \leq$ degree(z) Hence, it must be the case that v_k has not yet been enqueued.

If v_k was never enqueued, then v_{k-1} cannot have been enqueued and dequeued, as line 14-17 would have resulted in v_k being added to the queue. By the same argument above, $degree(v_{k-1}) \leq p$, hence it must be dequeued before z, so it must also have never been added to the queue. Continuing this argument in the same manner, it must be the case that all nodes in the path L were never placed onto the queue, as they all must have a degree less than p. But v_1 is a neighbor of the source node, s, by the definition of a promiscuous path. So, v_1 must be enqueued onto the queue during lines 3-6. Hence, this cannot be the case. Therefore it is impossible that z was dequeued before v_k was dequeued.

Assume Case 2 is true z was dequeued after v_k was dequeued and does not cause our algorithm to terminate at line 12. Therefore, it must be the case that $\tau(s, t, k)$ is set to a value greater than degree(z). But we have assumed v_k was dequeued. We know by definition of a promiscuous

path that $(v_k, t) \in E$, so line 24 must run when v_k is dequeued and set $\tau(s, t, k)$ to $p <$ degree(z). So it cannot be the case that z was dequeued after v_k was dequeued.

Hence, we have shown Lemma 1: no node with degree larger than p may be dequeued and not cause the algorithm to terminate. □

Lemma A.4. *The maximum number of nodes which may ever be placed onto the queue declared on line 2 of Algorithm 7 is* $\frac{b(p^k - 1)}{(p - 1)}$.

Proof. It is helpful to analyze the number of nodes on the queue in terms of their "depth" values. All neighbors of the source node are assigned a depth value of one when enqueued in lines 3-6. Each subsequent node is given a depth value of more than its parents depth value. No nodes may have a depth value greater than k as the only location where nodes with a depth value greater than one maybe enqueued are on lines 14-17, in this loop, we enqueue all neighbors, m, of the currently dequeued node, n onto the queue with a depth of n plus one. This loop may only run if the depth of n passes the check on line 13, which states that $depth_n < k$. We will now use induction to discuss the number of nodes which can be present on the queue for each depth value.

Base Case: The maximum number of neighbors of our source node is the branching factor of our graph b. Since the only way for a node to be enqueued with depth value one is the loop on lines 3-6. the maximum number of nodes on the queue with depth value one is $b * p^{1-1} = b$.

Inductive Step: Assume the queue has at most $b * (p^{i-1})$ nodes with depth number $i - 1$. In the worst case, each node with depth $i - 1$ will be dequeued. Additionally, in the worst case, every node that is dequeued will add the maximum number of nodes possible and continue execution of the algorithm (because by Lemma A.3, if a node has more than p neighbors, it cannot be dequeued without terminating the algorithm). As stated on line 15, each of these newly enqueued nodes will have depth $(i - 1) + 1$. Hence, in the worst case, the queue will have $b * (p^{i-1}) * p = b * p^i$ nodes with depth i.

Combining the fact that the maximum depth of nodes which can be placed on the queue is k and , the worst-case number of nodes on the queue will be:

$$\sum_{i=1}^{k} \text{\# nodes with depth i} = \sum_{i=1}^{k} b * p^{i-1} = \frac{b(p^k - 1)}{(p-1)}$$

□

Proof. We seek to prove Theorem A.3 (the Breadth First Search Variant of the promiscuity algorithm (Algorithm 7) is guaranteed to have a worst case runtime of $O(b * p^{k-1} * lg(b * p^{k-1}))$).

First, we must discuss the run times of a priority queue. We require only two priority queue operations for our algorithm, *Insert* and *Extract-Min*. Using a Fibonacci heap implementation, a priority queue can be implemented to have amortized run times of *Insert* in $\Theta(1)$ time and *Extract-Min* in $\Theta(lg(n))$ time (Cormen et al., 2009, Chapter 19). Since we are looking at the worst case run time over the entire execution of the algorithm, it is appropriate to use these amortized run times, as the priority queue will be accessed many times and it's execution time will perform as a summation of the average.

Next, we must discuss runtime of the construction of hash maps, also referred to as hash tables (Cormen et al., 2009, Chapter 11). In this scheme, we process keys we seek to insert and search through a *hash function* which spreads it evenly amongst a predefined search space. Under this scheme, with well chosen hash functions, we will have a search time of $O(1)$ and a memory requirement of $O(x)$ where x is the size of the search space. Additionally, with well chosen hash functions, this hash map will take $O(x)$ to construct.

Line 2 declares a priority queue and takes $O(1)$ time.

The loop in lines 3-6 of the BFS algorithm iterates by the number of neighbors of s, so it will execute degree(s) times. Lines 4 and 5 both declare variables and take $O(1)$ time. Line 6 requires us to insert into a priority queue, using a Fibonacci heap this will also be a $O(1)$ operation. The number of neighbors of s is constrained by the bounding factor b of the graph. Therefore the runtime of this loop is $O(b * (1 + 1 + 1))$ = $O(b)$.

Line 7 takes $O(1)$ time to declare a single variable.

The return statement on line 21 takes constant runtime.

We now seek to get analyze the runtime of the while loop on lines 9-20. First line 10 extracts the minimum node from the priority queue, this will take time $O(ln(x))$ where x is current size of the priority queue. We know by Lemma A.4 that the maximum number of nodes on the queue can be $\frac{b(p^k-1)}{(p-1)}$. So the worst case execution time for line 10 is $O(lg(\frac{b(p^k-1)}{(p-1)})) = O(lg(b*p^{k-1}))$. For convince, let us refer to the dequeued node as n. Lines 11-12 are both constant time operations. The check on line 13 is constant time. The loop on lines 14-17 follows the same execution pattern as lines 3-6, iterating over each neighbor of the dequeued node, with each operation on lines 15,16, and 17 taking constant time, therefore for any dequeued node n, the runtime of this loop will be $O(\text{degree}(n))$. We know from Lemma A.3 that all nodes on the queue must have degree less than or equal to p, so in the worst case this loop will take $O(p)$. The check on line 18 takes constant time. The performance of line 19 will depend on the specific hash map, but under well chosen hash functions this check will have runtime of $O(1)$. The update on line 20 has a constant runtime.

To help us further analyze the runtime of the while loop on lines 9-20, it is useful to calculate two values, the maximum number of nodes on the queue with depth $1, 2, \ldots, (k-1)$ and the number of nodes on the queue with depth k. Using the proof from Lemma A.4, we know each $\{$ # of nodes with depth i$\} = b * p^{i-1}$. So the number of nodes with depth $1, \ldots, (k-1)$ is $\sum_{i=1}^{k-1} b * p^{i-1} = \frac{b(p^{k-1}-1)}{(p-1)}$ and the maximum number of nodes with depth k is $b * p^{k-1}$.

The runtime of the loop for nodes with depth less than k requires the dequeuing of the priority queue on line 10, which takes $O(lg(b*p^{k-1}))$ time. It will pass the check on line 13 requires execution of the loop on lines 14-17. This loop requires $O(p)$ execution time for each node dequeued. So the contribution to the runtime from these nodes is

$$
\begin{aligned}
&O(\{\text{ \# of nodes with depth } 1, 2, \ldots, (k-1)\} * \{\text{execution time of while loop}\}) \\
=&O((\frac{b(p^{k-1}-1)}{(p-1)}) * (p + lg(b * p^{k-1}))) \\
=&O((b * p^{k-2}) * (p + lg(b * p^{k-1})) \\
=&O(b * p^{k-1} + b * p^{k-2} * lg(b * p^{k-1}))
\end{aligned}
\tag{A.3}
$$

The runtime of the loop for nodes with depth k still requires the dequeuing of the priority queue on line 10, which takes $O(lg(b * p^{k-1}))$ time. It will fail the check on line 13, but pass the check on line 18. So we must execute lines 19, 20, which both take constant time. So the contribution to the runtime from these nodes is

$$
\begin{aligned}
&O(\{\text{ \# of nodes with depth } k\} * \{\text{execution time of while loop}\}) \\
=&O((b(p^{k-1})) * (1 + lg(b * p^{k-1}))) \\
=&O(b * p^{k-1} * lg(b * p^{k-1}))
\end{aligned}
\tag{A.4}
$$

So the final runtime of the while loop on lines 9-20 is

$$
\begin{aligned}
&O(\{\text{ \# of nodes with depth } 1, 2, \ldots, (k-1)\} * \{\text{execution time of while loop}\}) \\
+&O(\{\text{ \# of nodes with depth } k\} * \{\text{execution time of while loop}\}) \\
=&O(b * p^{k-1} + b * p^{k-2} * lg(b * p^{k-1})) + O(b * p^{k-1} * lg(b * p^{k-1})) \\
=&O(b * p^{k-1} + b * p^{k-2} * lg(b * p^{k-1}) + b * p^{k-1} * lg(b * p^{k-1})) \\
=&O(b * p^{k-1} * lg(b * p^{k-1}))
\end{aligned}
\tag{A.5}
$$

So the final worst case execution of the algorithm will be constant time operations from lines 2, 7, 21, $O(b)$ execution time for the loop on lines 3-6, $O(b)$ execution time for constructing

the hashmap on line 8, and $O(b * p^{k-1} * lg(b * p^{k-1}))$ execution time for the loop on lines 9-20. So the final execution time will be

$$\begin{aligned} & O(1 + 1 + 1 + b + b + b * p^{k-1} * lg(b * p^{k-1})) \\ =& O(b * p^{k-1} * lg(b * p^{k-1})) \end{aligned} \tag{A.6}$$

where most of the execution time will be dominated by the processing of nodes of depth k within the while loop. □

A.1.4 Depth First Search Promiscuity Value Algorithm

We now seek to discuss the other variant of the promiscuity algorithm, Algorithm 8. In this version of the algorithm, we prioritize nodes with higher *depth* values for exploration first; when nodes have tied depth values, the node with the lower degree should be chosen. This variant of the algorithm will have a higher runtime in the majority of cases, but has lower complexity in memory usage which may make it attractive in certain cases. This version has also been implemented, as described in the Implementation section (Section A.3).

Lemma A.5. *No node with depth of one AND degree larger than p, may be dequeued and not cause the algorithm to terminate on line 12. Where p is the promiscuity value (as defined in Definition A.3) for source s and tail t over graph G with path length k and depth being the length of path.*

Proof. Proof by contradiction. Let us assume it is the case that a node, w_1, is dequeued with depth of one, and $degree(w_1) > p$. We know that by definition of promiscuity score (Definition A.4) there is some path $L = \{v_1, v_2, \ldots, v_k\}$ such that the promiscuity value (Definition A.3) $\tau(s, t, k) = promiscuity(L)$. It must be the case that since both v_1 and w_1 have a depth value of one, they are neighbors of s. So these nodes will be placed onto the queue in the for loop on lines 3-6.

Algorithm 8: Depth First Search Algorithm for finding the promiscuity score $\tau(s, t, l)$.

Data: Graph G, source node s and target node t in G, path length k
Result: The promiscuity score $\tau(s, t, k)$ from s to t over graph G

```
1  begin
2      Q ← PriorityQueue(); // min priority queue keyed on score_n
3      for n ∈ neighbors(s) do
4          depth_n ← 1;
5          score_n ← degree(n);
6          Q.append(depth_n, score_n, n);
7      τ(s, t, k) ← ∞;
8      t_neighbors ← BuildHashMap(neighbors(t));
9      while Q ≠ ∅ do
10         depth_n, score_n, n ← Q.pop();
           // Gets node on queue with lowest promiscuity score.
11         if degree(n) > τ(s, t, k) then
12             return τ(s, t, k) ;
13         if depth_n < k then
14             for m ∈ neighbors(n) do
15                 depth_m ← depth_n + 1;
16                 score_m ← MAX{score_n, degree(m)};
17                 Q.append(depth_m, score_m, m);
18         if depth_n == k then
19             if n ∈ t_neighbors then
20                 τ(s, t, k) ← MIN{τ(s, t, k), score_n};
21     return τ(s, t, k);
```

We'll now show that Algorithm 8 must dequeue and execute on all nodes in path L before dequeuing w_1 through induction.

Base Case: It must be the case that node v_1 must be dequeued before node w_1 by line 10. This is because as $degree(v_1) \leq promiscuity(L)$, so $degree(v_1) \leq p$ which entails that $degree(v_1) < degree(w_1)$. Since $depth_{v_1} < k$ and v_2 is a neighbor of v_1, line 13 will be passed and v_2 will be added onto the queue with depth 2 in the for loop on lines 14-17.

Inductive Case: Assume node v_i with $depth_{v_i} = i \geq 2$ has been placed onto the queue. It must be the case that this node is dequeued before w_1 is dequeued. This is because $depth_{v_i} >$

$depth_{w_1}$, which by construction of our priority queue, means it gets priority. It must be the case that either (1) $depth_{v_i} < k$ or (2) $depth_{v_i} = k$.

Case (1): If $depth_{v_i} < k$, since node v_{i+1} is a neighbor of v_i, in the for loop on lines 14-17 v_{i+1} will be placed onto the queue with $depth_{v_{i+1}} = i + 1$.

Case (2): If $depth_{v_i} = k$, the check on line 18 will be passed, additionally, since we know L is a path from s to t, the check on line 19 will also pass. So finally line 20 will be called and update the currently tracked value $\tau(s, t, k)$ to p.

Since all nodes in path L must be dequeued before w_1 can be dequeued, this will result in $\tau(s, t, k)$ being set to p. In this case, when w_1 is dequeued, when the check on line 11 is reached, it will succeed and line 12 will be executed, terminating the algorithm.

□

Lemma A.6. *It is impossible for two nodes to be placed onto the queue with any depth value i and with different parent nodes (the parent is the node which was dequeued when the node was placed onto the queue).*

Proof. All nodes with depth one must be placed on the queue by the source node s. Let us assume it is the case that depth $i \geq 2$.

Proof by contradiction: Let it be the case that was have two nodes on the queue v_i, w_i, with depth i. Let it also be the case that v_i has parent v_{i-1} and w_i has parent w_{i-1} and $v_{i-1} \neq w_{i-1}$ □

Lemma A.7. *The maximum number of nodes which may be placed onto the priority queue in Algorithm 8 with depth value of one is b.*

Proof. The only place we may place nodes with a depth value of one onto the queue is in the loop in lines 3-6. The number of nodes which may be placed onto the queue is bound by the number of neighbors of s. In the worst case, s will have a degree of b. Therefore the maximum number of nodes with depth one that can be placed on the queue is b. □

Lemma A.8. *The maximum number of nodes which may be placed onto the priority queue at one time in Algorithm 8 with depth value of 2 is p.*

Proof. Using the statement of Lemma A.5 (No node with depth of one AND degree larger than p, may be dequeued and not cause the algorithm to terminate on line 12). Let us assume we know that each node of depth one which is dequeued in the while loop on lines 9-20. The only place nodes of depth 2 may be added to the queue is on line 17. This line will run for each neighbor of our dequeued node, which we know to have degree less than or equal to p. Additionally by lemma A.6, we know that it is impossible for there to be two different parents to place nodes with depth 2 on the queue. So there may only be p nodes with depth 2 on the queue at all times. □

Lemma A.9. *The maximum number of nodes which may be placed onto the priority queue in Algorithm 8 with depth value of* $\{3, 4, \ldots, k\text{-}1, k\}$ *is* b.

Proof. The argument here is the same argument as in Lemma A.8. Except it does not have any restriction on number of nodes is the bounding factor b. Let us assume a node u with depth i is dequeued in the while loop on lines 9-20. We know that u must have maximal degree b. So each neighbor of u will be enqueued with depth $i + 1$, which means at most b nodes with depth $i + 1$ will be enqueued. No further nodes of depth $i + 1$ can be enqueued until all nodes of this depth have been dequeued and processed. This is because when we dequeue a node on line 10, we dequeue nodes of higher degree first, and by line 15, all nodes enqueued must have a depth value greater than the node currently dequeued. □

Lemma A.10. *The maximum number of nodes which may be placed onto the priority queue in Algorithm 8 is* $O((k-1) * b + p)$.

Proof. The statement of lemma A.10 follows from the statements of Lemma A.7, Lemma A.8, and Lemma A.9. Knowing that no nodes with depth value greater than k may ever be placed on the queue, we simply sum the maximum number of nodes which may be placed on the queue for each depth value. Thus we get

$$
\begin{aligned}
&\{\text{Maximum \# of nodes on the queue}\} \\
=&\{\text{Maximum \# of nodes on the queue with depth } 1\} \\
+&\{\text{Maximum \# of nodes on the queue with depth } 2\} \\
+&\{\text{Maximum \# of nodes on the queue with depth } 3, 4, \ldots, k\} \\
=&b + p + (k-2) * b \\
=&(k-1) * b + p
\end{aligned}
\tag{A.7}
$$

□

Lemma A.11. *The maximum number of nodes of depth i that may be explored (enqueued and subsequently dequeued) is $O(p * b^{i-1})$-1 (for i greater than or equal to 2).*

Proof. Proof by induction. Base case. Let depth value $i = 2$. As we have discussed in lemma A.7, lemma A.8, and lemma A.9, we know the number of nodes with depth 1 which may be placed onto the queue is b. Additionally, we know that each of those nodes may be dequeued and place p nodes of depth 2 onto the queue (which would be $b * p$ total nodes). Since we can establish no restriction on the degree size of nodes depth 2, each of these nodes may place b nodes onto the queue. So we would in the worst case see $O(p * b) = O(p * b^{2-1}$ nodes of depth 2 placed on the queue.

Inductive step. Let's assume we place and explore $O(b^{i-2} * p)$ nodes of depth $i - 1$ on the queue. Each of these nodes may place b nodes onto the queue with depth i, which would need to be explored. Therefore we would have $O(b * b^{i-2} * p) = O(b^{i-1} * p)$ nodes of depth i placed and explored on the queue. □

Theorem A.4. *The runtime of the DFS variant of the Promiscuity Algorithm (Algorithm 8) is $O(b^{k-1} * p * lg(b * (k-1) + p))$*

Proof. This argument has a very similar structure to that of Theorem A.3. First it is important to assume we use a Fibonacci heap for the priority queue implementation, which can be implemented

to have amortized run times of *Insert* in $\Theta(1)$ time and *Extract-Min* in $\Theta(lg(n))$ time (Cormen et al., 2009, Chapter 19).

Similarly we must look at the time to build and construct hash tables/hash maps. We know the time to construct hashmap is $O(x)$ where x is the size of the search space, and with a well chosen hash function, the search time for the hashmap will be $O(1)$ (Cormen et al., 2009, Chapter 11).

Line 2 declares a priority queue and will take $O(1)$ time.

The loop on lines 3-6 will be executed for each neighbor of s, so $degree(s)$ times. Lines 4 and 5 are both variable declarations and take $O(1)$ time. Line 6 will take $O(1)$ to execute on a Fibonacci heap. So this loop will take $O(degree(s)) = O(b)$ time to execute.

Line 7 is a variable declaration and takes $O(1)$ time.

Line 8 is requires building a hashmap for each neighbor of t, which will take $O(degree(t)) = O(b)$ time to execute.

The return statement on line 21 will take constant time to execute.

So the runtime for all lines not within the while loop is

$$\begin{aligned} &O(\{\text{runtime of line 2}\}) + O(\{\text{runtime of lines 3-6}\}) \\ &+O(\{\text{runtime of line 8}\}) + O(\{\text{runtime of line 21}\}) \\ &=O(1) + O(b) + O(b) + O(1) \\ &=O(b) \end{aligned}$$

Now we shall analyze the runtime of the while loop on lines 9-20.

Line 10 will pop the current top node of the priority queue. This will take $\Theta(\{\text{size of queue}\})$. We know by lemma A.10 that the maximum size of the priority queue is $O(b * (k-1) + p)$. So the runtime of this line is $O(lg(b * (k-1) + p))$.[2]

Line 11 is a comparison which will take $O(1)$ time.

Line 12 is a return statement which will take $O(1)$ time.

2. We could do a more nuanced analysis of the runtime of line 10 based on the depth levels of the nodes we're exploring, as the queue will only have $O(p + (k-1) * b)$ nodes when we are exploring nodes at depth k, but accounting for this does not affect our final result.

Line 13 is a comparison which will take $O(1)$ time.

Let us refer to the dequeued node as n. The for loop on lines 14-17 will only execute for dequeued nodes with depth values $depth_n \in \{1, 2, \ldots, k-1\}$. It will execute for each neighbor n. Line 15 is a variable creation and will take $O(1)$ time. Line 16 is a maximum operation which takes $O(1)$ time. Line 17 is an append which takes $O(1)$ time. So this loop will take $O(degree(n) * 1) = O(b)$ time to execute.

Line 18 is a comparison which will take $O(1)$ time to compute.

Lines 19 and 20 will only execute for nodes with $depth_n = k$. Line 19 will take $O(1)$ time to execute with a well constructed hashmap. Line 20 is a variable update which takes $O(1)$ time to execute.

Combining all the above statements, for nodes with depth $\in \{1, 2, \ldots, k-1\}$ the while loop on lines 9-20 will take

$$
\begin{aligned}
&O(\{\text{Runtime of line 10}\}) + O(\{\text{Runtime of line 11}\}) + O(\{\text{Runtime of line 12}\}) \\
&+O(\{\text{Runtime of line 13}\}) + O(\{\text{Runtime of lines 14-17}\}) + O(\{\text{Runtime of line 18}\}) \\
&=O(lg(p + b * (k-1))) + O(1) + O(1) \\
&+O(1) + O(b) + O(1) \\
&=O(lg(b * (k-1) + p) + b)
\end{aligned}
\tag{A.8}
$$

And for nodes with depth$= k$ the while loop on lines 9-20 will take

$$
\begin{aligned}
&O(\{\text{Runtime of line 10}\}) + O(\{\text{Runtime of line 11}\}) + O(\{\text{Runtime of line 12}\}) \\
&+O(\{\text{Runtime of line 13}\}) + O(\{\text{Runtime of line 18}\}) + O(\{\text{Runtime of lines 19-20}\}) \\
&=O(lg(p + b * (k-1))) + O(1) + O(1) \\
&+O(1) + O(1) + O(1) \\
&=O(lg(b * (k-1) + p))
\end{aligned} \tag{A.9}
$$

We'll now count how many nodes we explore at different depths.

We explore at most $O(b)$ nodes with depth 1.

We must now look at how many nodes of depths $\in \{2, \ldots, k-1\}$ we may explore in the execution of this Algorithm. Looking at the result of lemma A.11, we know for depth value i, the maximum number of nodes we may explore is $O(p * b^{i-1})$. So the number of nodes we explore is

$$
\sum_{i=2}^{k-1} O(p * b^{i-1}) = \sum_{i=1}^{k-2} O(p * b^{i}) = O(\frac{p(b^k - b^2)}{(b-1)b}) = O(p * b^{k-2})
$$

So combining the two statements above, for depths $\in \{1, 2, \ldots, k-1\}$ we explore $O(\{\text{Nodes with depth 1}\}) + O(\{\text{Nodes with depths} \in \{2, \ldots, k-1\}\}) = O(b) + O(p * b^{k-2}) = O(p * b^{k-2})$ nodes.

By lemma A.11 we know the maximum number of nodes of depth k we can explore is $O(p * b^{k-1})$.

So the runtime of the while loop on lines 9-20 will take

$$
\begin{aligned}
& O(\{\text{Number nodes explored of depth} \in \{1, 2, \ldots, k-1\}\}) \\
& O(\{\text{Runtime for nodes of depth} \in \{1, 2, \ldots, k-1\}\}) \\
+ & O(\{\text{Number nodes explored of depth}= k\}) \\
* & O(\{\text{Runtime for nodes of depth}= k\}) \\
= & O(p * b^{k-2}) \\
* & O(lg(p + b * (k-1)) + b) \text{(using Eq A.8)} \\
+ & O(p * b^{k-1}) \\
* & O(lg(p + b * (k-1))) \text{(using Eq A.9)} \\
= & O(p * b^{k-2} * lg(b * (k-1) + p) + p * b^{k-1}) \text{(runtime for handling all nodes with depths 1-(k-1))} \\
+ & O(p * b^{k-1} * lg(b * (k-1) + p)) \text{(runtime for handling all nodes with depth k)} \\
= & O(p * b^{k-2} * lg(b * (k-1) + p) + p * b^{k-1} + p * b^{k-1} * lg(b * (k-1) + p)) \\
= & O(p * b^{k-1} * lg(b * (k-1) + p))
\end{aligned}
\tag{A.10}
$$

Thus we find the desired result. Here we notice the same phenomenon as we saw in the runtime of the BFS Algorithm (Algorithm 7), where the runtime is completely dominated by the the nodes of the greatest depth.

□

A.2 Promiscuity Worst Case Memory Usage and Runtime Summary

Here we seek to summarize the results from the previous sections.

In Table A.1 we summarize all of the results for runtime and memory usage of all three approaches we described.

In comparing the results we see some interesting results. In the case where no path can be found, Algorithm 6, Algorithm 7 and Algorithm 8 reduce to a very poor runtime ($O(b^k * lg(b^k))$),

Algorithm	Runtime	Memory Usage
Naive Approach (Algorithm 6)	$O(b^k * lg(b))$	$O(b^k)$
BFS Optimized Approach (Algorithm 7)	$O(b * p^{k-1} * lg(b * p^{k-1}))$	$O(b * p^{k-1})$
DFS Optimized Approach (Algorithm 8)	$O(b^{k-1} * p * lg(p + (k-1)b))$	$O(b * (k-1) + p)$

Table A.1: A summary of runtimes and memories for the various algorithms for calculating promiscuity.

which is actually worse than the naive approach. This is due to the fact that while both algorithms would need check all possible paths between the source and tail, Algorithm 7 must take more time constructing, and dequeuing nodes from the priority queue.

A.3 Implementation

Neo4J is the underlying graph database software which powers multiple of the largest biomedical knowledge graphs. This software library enables users to create large queryable data structures. Both ROBOKOP (Section 1.5.1) and HetioNet (Section 1.5.2) have Neo4J deployments of their knowledge graphs.

Neo4J enables users to extend the database software with custom code. This process is analogous to the "User-defined Functions" (UDFs) commonly utilized in the relational database community for SQL. In Neo4J the custom user code are called **Procedures**. These procedures can be directly invoked utilizing the Cypher language for any server. The process of installing plugins to servers is simple and flexible, as it takes advantage of the interoperability of the Java code base.

To integrate any plugin into a new database, a user simply needs to download the source code for the algorithm (`https://github.com/DnlRKorn/promiscuity-procedure`) and compile it using any Java SDK to a JAR file. The JAR file must then be inserted into the $NEO4J_HOME/plugins directory, and the server restarted.

BIBLIOGRAPHY

Adjei, A. A. (2004). Pharmacology and mechanism of action of pemetrexed. *Clinical lung cancer*, 5:S51–S55.

Ahalt, S. (2017). Democratized data driving discovery: Renci strategic plan.

Akujuobi, U., Spranger, M., Palaniappan, S. K., and Zhang, X. (2020). T-pair: Temporal node-pair embedding for automatic biomedical hypothesis generation. *IEEE Transactions on Knowledge and Data Engineering*, page 1–1.

Alves, V. M., Korn, D., Pervitsky, V., Thieme, A., Capuzzi, S. J., Baker, N., Chirkova, R., Ekins, S., Muratov, E. N., Hickey, A., and Tropsha, A. (2022). Knowledge-based approaches to drug discovery for rare diseases. *Drug Discovery Today*, 27(2):490–502.

Amberger, J. S., Bocchini, C. A., Schiettecatte, F., Scott, A. F., and Hamosh, A. (2015). Omim.org: Online mendelian inheritance in man (omim®), an online catalog of human genes and genetic disorders. *Nucleic Acids Research*, 43(D1):D789–D798.

Amberger, J. S., Bocchini, C. A., Scott, A. F., and Hamosh, A. (2019). Omim.org: leveraging knowledge across phenotype–gene relationships. *Nucleic Acids Research*, 47(D1):D1038–D1043.

Andreini, C., Cavallaro, G., Lorenzini, S., and Rosato, A. (2013). Metalpdb: A database of metal sites in biological macromolecular structures. *Nucleic Acids Research*, 41(D1).

Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., and Vrgoč, D. (2017). Foundations of modern query languages for graph databases. *ACM Computing Surveys*, 50(5).

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., and et al. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (2012). *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media.

Austin, C. P. (2016). Ncats strategic plan.

Avram, S., Bologa, C. G., Holmes, J., Bocci, G., Wilson, T. B., Nguyen, D. T., Curpan, R., Halip, L., Bora, A., Yang, J. J., and et al. (2021). Drugcentral 2021 supports drug discovery and repositioning. *Nucleic Acids Research*, 49(D1):D1160–D1169.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Inc., USA.

Bailey, C. J. (2017). Metformin: historical overview. *Diabetologia*, 60(9):1566–1576.

Bakal, G., Talari, P., Kakani, E. V., and Kavuluru, R. (2018). Exploiting semantic patterns over biomedical knowledge graphs for predicting treatment and causative relations. *Journal of Biomedical Informatics*, 82:189–199.

Baker, N. C. and Hemminger, B. M. (2010). Mining connections between chemicals, proteins, and diseases extracted from medline annotations. *Journal of biomedical informatics*, 43(4):510–519.

Bakken, S. (2020). Informatics is a critical strategy in combating the covid-19 pandemic. *Journal of the American Medical Informatics Association: JAMIA*, 27(6):843–4.

Banda, J. M., Evans, L., Vanguri, R. S., Tatonetti, N. P., Ryan, P. B., and Shah, N. H. (2016). A curated and standardized adverse drug event resource to accelerate drug safety research. *Scientific data*, 3:160026.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Barabási, A.-L. (2016). Network Science. In *Network Science*, chapter 4.2 Power Laws and Scale-Free Networks, pages 112–115. Cambridge University Press, Cambridge, United Kingdom, 1st edition.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483.

Barceló, P. (2013). Querying graph databases. In *Proceedings of the 32nd symposium on Principles of database systems - PODS '13*, page 175, New York, New York, USA. ACM Press.

Barceló, P., Libkin, L., and Reutter, J. L. (2014). Querying Regular Graph Patterns. *Journal of the ACM*, 61(1):1–54.

Bateman, A. (2019). Uniprot: A worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*.

Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., and Morissette, J. (2008). Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706–716.

Ben-Menachem, E. (2004). Pregabalin pharmacology and its relevance to clinical practice. *Epilepsia*, 45:13–18.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific american*, 284(5):34–43.

Bi, Y., Might, M., Vankayalapati, H., and Kuberan, B. (2017). Repurposing of proton pump inhibitors as first identified small molecule inhibitors of endo-β-n-acetylglucosaminidase (engase) for the treatment of ngly1 deficiency, a rare genetic disease. *Bioorganic & medicinal chemistry letters*, 27(13):2962–2966.

Binder, J., Ursu, O., Bologa, C., Jiang, S., Maphis, N., Dadras, S., Chisholm, D., Weick, J., Myers, O., Kumar, P., Yang, J. J., Bhaskar, K., and Oprea, T. I. (2022). Machine learning prediction and tau-based screening identifies potential Alzheimer's disease genes relevant to immunity. *Communications Biology*, 5(1):125.

Binns, D., Dimmer, E., Huntley, R., Barrell, D., O'Donovan, C., and Apweiler, R. (2009). Quickgo: A web-based tool for gene ontology searching. *Bioinformatics*, 25(22):3045–3046.

Bizon, C., Cox, S., Balhoff, J., Kebede, Y., Wang, P., Morton, K., Fecho, K., and Tropsha, A. (2019). Robokop kg and kgb: Integrated knowledge graphs from federated sources. *Journal of Chemical Information and Modeling*, 59(12):4968–4973.

Bjornsson, A. S., Didie, E. R., and Phillips, K. A. (2010). Body dysmorphic disorder. *Dialogues in clinical neuroscience*, 12(2):221.

Blatt, J., Farag, S., Corey, S. J., Sarrimanolis, Z., Muratov, E., Fourches, D., Tropsha, A., and Janzen, W. P. (2014). Expanding the scope of drug repurposing in pediatrics: The children's pharmacy collaborativetm. *Drug discovery today*, 19(11):1696–8.

Block, G., Liss, C., Reines, S., Irr, J., and Nibbelink, D. (1997). Comparison of immediate-release and controlled release carbidopa/levodopa in parkinson's disease. *European neurology*, 37(1):23–27.

Bobrowski, T., Chen, L., Eastman, R. T., Itkin, Z., Shinn, P., Chen, C. Z., Guo, H., Zheng, W., Michael, S., Simeonov, A., and et al. (2021). Synergistic and antagonistic drug combinations against sars-cov-2. *Molecular Therapy*, 29(2):873–885.

Bobrowski, T., Melo-Filho, C. C., Korn, D., Alves, V. M., Popov, K. I., Auerbach, S., Schmitt, C., Moorman, N. J., Muratov, E. N., and Tropsha, A. (2020). Learning from history: do not flatten the curve of antiviral research! *Drug discovery today*, 25(9):1604–1613.

Boolell, M., Allen, M. J., Ballard, S. A., Gepi-Attee, S., Muirhead, G. J., Naylor, A. M., Osterloh, I. H., and Gingell, C. (1996). Sildenafil: an orally active type 5 cyclic gmp-specific phosphodiesterase inhibitor for the treatment of penile erectile dysfunction. *International journal of impotence research*, 8(2):47–52.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 26, USA. Curran Associates, Inc.

Broido, A. D. and Clauset, A. (2019). Scale-free networks are rare. *Nature communications*, 10(1):1–10.

Brouillette, M. (2019). Ai added to the curriculum for doctors-to-be. *Nature Medicine*, 25(12):1808–1809.

Buchdunger, E., Cioffi, C. L., Law, N., Stover, D., Ohno-Jones, S., Druker, B. J., and Lydon, N. B. (2000). Abl protein-tyrosine kinase inhibitor sti571 inhibits in vitro signal transduction mediated by c-kit and platelet-derived growth factor receptors. *The Journal of pharmacology and experimental therapeutics*, 295(1):139–45.

Bult, C. J., Blake, J. A., Smith, C. L., Kadin, J. A., Richardson, J. E., Anagnostopoulos, A., Asabor, R., Baldarelli, R. M., Beal, J. S., Bello, S. M., and et al. (2019). Mouse genome database (mgd) 2019. *Nucleic Acids Research*, 47(D1):D801–D806.

Buniello, A., Macarthur, J. A., Cerezo, M., Harris, L. W., Hayhurst, J., Malangone, C., McMahon, A., Morales, J., Mountjoy, E., Sollis, E., and et al. (2019). The nhgri-ebi gwas catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic Acids Research*, 47(D1):D1005–D1012.

Cahill, K. N., Katz, H., Cui, J., Lai, J., Kazani, S., Crosby-Thompson, A., Garofalo, D., Castro, M., Jarjour, N. N., DiMango, E., and et al. (2017a). Effect of kit inhibition by imatinib on airway mast cells in patients with severe refractory asthma (kia). *Journal of Allergy and Clinical Immunology*, 139(2):AB169.

Cahill, K. N., Katz, H. R., Cui, J., Lai, J., Kazani, S., Crosby-Thompson, A., Garofalo, D., Castro, M., Jarjour, N., DiMango, E., and et al. (2017b). Kit inhibition by imatinib in patients with severe refractory asthma. *New England Journal of Medicine*, 376(20):1911–1920.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.

Calcaterra, N. E. and Barrow, J. C. (2014). Classics in chemical neuroscience: diazepam (valium). *ACS chemical neuroscience*, 5(4):253–260.

Can, T. (2014). Introduction to bioinformatics. *Methods in molecular biology (Clifton, N.J.)*, 1107:51–71.

Canese, K. and Weis, S. (2013). *PubMed: The bibliographic database*, page 13–24. National Center for Biotechnology Information Bethesda, MD.

Capdeville, R., Buchdunger, E., Zimmermann, J., and Matter, A. (2002). Glivec (sti571, imatinib), a rationally developed, targeted anticancer drug. *Nature reviews. Drug discovery*, 1(7):493–502.

Capuzzi, S. J., Thornton, T. E., Liu, K., Baker, N., Lam, W. I., O'Banion, C. P., Muratov, E. N., Pozefsky, D., Tropsha, A., O'Banion, C. P., and et al. (2018). Chemotext: A publicly available web server for mining drug–target–disease relationships in pubmed. *Journal of Chemical Information and Modeling*, 58(2):212–218.

Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., Lewis, S., Hub, t. A., and Group, t. W. P. W. (2009). Amigo: online access to ontology and annotation data. *Bioinformatics*, 25(2):288–289.

Chambers, J., Davies, M., Gaulton, A., Hersey, A., Velankar, S., Petryszak, R., Hastings, J., Bellis, L., McGlinchey, S., and Overington, J. P. (2013). Unichem: A unified chemical structure cross-referencing and identifier tracking system. *Journal of Cheminformatics*, 5(1).

Chen, B., Dong, X., Jiao, D., Wang, H., Zhu, Q., Ding, Y., and Wild, D. J. (2010a). Chem2bio2rdf: A semantic framework for linking and data mining chemogenomic and systems chemical biology data. *BMC Bioinformatics*, 11.

Chen, J.-K., Shen, C.-R., and Liu, C.-L. (2010b). N-acetylglucosamine: production and applications. *Marine drugs*, 8(9):2493–516.

Chen, M., Tian, Y., Yang, M., and Zaniolo, C. (2017). Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. *IJCAI International Joint Conference on Artificial Intelligence*, 0:1511–1517.

Chervitz, S. A., Deutsch, E. W., Field, D., Parkinson, H., Quackenbush, J., Rocca-Serra, P., Sansone, S.-A., Stoeckert Jr, C. J., Taylor, C. F., Taylor, R., and et al. (2011). Data standards for omics data: the basis of data sharing and reuse. *Methods in molecular biology (Clifton, N.J.)*, 719:31–69.

Conti-Fine, B. M., Milani, M., Kaminski, H. J., et al. (2006). Myasthenia gravis: past, present, and future. *The Journal of clinical investigation*, 116(11):2843–2854.

Coordinators, N. R. (2016). Database resources of the national center for biotechnology information. *Nucleic acids research*, 44(D1):D7–D19.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Corsello, S. M., Bittker, J. A., Liu, Z., Gould, J., McCarren, P., Hirschman, J. E., Johnston, S. E., Vrcic, A., Wong, B., Khan, M., et al. (2017). The drug repurposing hub: a next-generation drug library and information resource. *Nature medicine*, 23(4):405–408.

Culmone, R., Falcioni, M., Giuliodori, P., Merelli, E., Orru, A., Quadrini, M., Ciampolini, P., Grossi, F., and Matrella, G. (2014). AAL domain ontology for event-based human activity recognition. In *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6.

Dai, J., Liu, M., Ai, Q., Lin, L., Wu, K., Deng, X., Jing, Y., Jia, M., Wan, J., and Zhang, L. (2014). Involvement of catalase in the protective benefits of metformin in mice with oxidative liver injury. *Chemico-Biological Interactions*, 216(1):34–42.

Davis, A. P., Grondin, C. J., Johnson, R. J., Sciaky, D., Wiegers, J., Wiegers, T. C., and Mattingly, C. J. (2021). Comparative toxicogenomics database (ctd): update 2021. *Nucleic acids research*, 49(D1):D1138–D1143.

Dell'Anno, I., Martin, S. A., Barbarino, M., Melani, A., Silvestri, R., Bottaro, M., Paolicchi, E., Corrado, A., Cipollini, M., Melaiu, O., et al. (2021). Drug-repositioning screening identified fludarabine and risedronic acid as potential therapeutic compounds for malignant pleural mesothelioma. *Investigational new drugs*, 39(3):644–657.

Denton, N., Molloy, M., Charleston, S., Lipset, C., Hirsch, J., Mulberg, A. E., Howard, P., and Marsh, E. D. (2021). Data silos are undermining drug development and failing rare disease patients. *Orphanet Journal of Rare Diseases*, 16(1):161.

Desai, R. J., Varma, V. R., Gerhard, T., Segal, J., Mahesri, M., Chin, K., Nonnenmacher, E., Gabbeta, A., Mammen, A. M., Varma, S., and et al. (2020). Targeting abnormal metabolism in alzheimer's disease: The drug repurposing for effective alzheimer's medicines (dream) study. *Alzheimer's & dementia (New York, N. Y.)*, 6(1):e12095.

Devore, J. L. (2015). *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 9th edition.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 601–610.

Dong, Y., Chawla, N. V., and Swami, A. (2017). Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 135–144, New York, NY, USA. Association for Computing Machinery.

Doğan, T., Atas, H., Joshi, V., Atakan, A., Rifaioglu, A., Nalbat, E., Nightingale, A., Saidi, R., Volynkin, V., Zellner, H., Cetin-Atalay, R., Martin, M., and Atalay, V. (2021). CROssBAR: comprehensive resource of biomedical relations with knowledge graph representations. *Nucleic Acids Research*, 49(16):e96–e96. gkab543.

Drozd, A., Gladkova, A., and Matsuoka, S. (2016). Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, page 3519–3530.

DrugBank (2021). Famotidine — drugbank entry.

Dwyer, D. S., Donohoe, D., Lu, X.-H., and Aamodt, E. J. (2005). Mechanistic connections between glucose/lipid disturbances and weight gain induced by antipsychotic drugs. *International review of neurobiology*, 65:211–247.

Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *CEUR Workshop Proceedings*, 1695.

Ekins, S. and Perlstein, E. O. (2018). Doing it all - how families are reshaping rare disease research. *Pharmaceutical Research*, 35(10):192.

Federal Drug Administration (2018). Developing products for rare diseases and conditions.

Feilmayr, C. and Wöß, W. (2016). An analysis of ontologies and their success factors for application to business. *Data & Knowledge Engineering*, 101:1–23.

Fenner, F. (1982). Global eradication of smallpox. *Reviews of infectious diseases*, 4(5):916–930.

Field, S. L. (1991). Why your friends have more friends than you do. *American Journal of Sociology*, 96(6):1464–1477.

Finsterer, J. (2003). Ptosis: causes, presentation, and management. *Aesthetic plastic surgery*, 27(3):193–204.

Fleming, A. (1929). On the antibacterial action of cultures of a penicillium, with special reference to their use in the isolation of b. influenzae. *British journal of experimental pathology*, 10(3):226.

for Advancing Translational Sciences, N. C. (2017). Biomedical data translator: Technical feasibility assessment of reasoning tool (ot2).

Frampton, J. E. and Foster, R. H. (2005). Pregabalin in the treatment of postherpetic neuralgia. *Drugs*, 65(1):111–118.

Franke, M., Desai, S., Deng, Q., Wellsandt, S., Hribernik, K. A., and Thoben, K. D. (2018). Semantic web: Ontological search approach. *CEUR Workshop Proceedings*, 2198.

Freedberg, D. E., Conigliaro, J., Wang, T. C., Tracey, K. J., Callahan, M. V., Abrams, J. A., Group, F. R. G. F. R., Sobieszczyk, M. E., Markowitz, D. D., Gupta, A., and et al. (2020). Famotidine use is associated with improved clinical outcomes in hospitalized covid-19 patients: A propensity score matched retrospective cohort study. *Gastroenterology*, 159(3):1129–1131.

Frezza, A. M., Botta, L., Trama, A., Dei Tos, A. P., and Stacchiotti, S. (2019). Chordoma: Update on disease, epidemiology, biology and medical therapies. *Current Opinion in Oncology*, 31(2):114–120.

Fuehrer, N. E., Marchevsky, A. M., and Jagirdar, J. (2009). Presence of c-kit-positive mast cells in obliterative bronchiolitis from diverse causes. *Archives of pathology & laboratory medicine*, 133(9):1420–5.

Galiè, N., Ghofrani, H. A., Torbicki, A., Barst, R. J., Rubin, L. J., Badesch, D., Fleming, T., Parpia, T., Burgess, G., Branzi, A., and et al. (2005). Sildenafil citrate therapy for pulmonary arterial hypertension. *New England Journal of Medicine*, 353(20):2148–2157.

Gallagher, B. (2006). Matching structure and semantics: A survey on graph-based pattern matching. In *AAAI Fall Symposium: Capturing and Using Patterns for Evidence Detection*, volume 45, Menlo Park, California.

Gandomi, A. and Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35:8.

Ganti, V. (2009). *Data Cleaning*, pages 561–564. Springer US, Boston, MA.

Gardner, M., Talukdar, P., Krishnamurthy, J., and Mitchell, T. (2014). Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406, Doha, Qatar. Association for Computational Linguistics.

Gardner, M., Talukdar, P. P., Kisiel, B., and Mitchell, T. (2013). Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838, Seattle, Washington, USA. Association for Computational Linguistics.

Gaulton, A., Hersey, A., Nowotka, M., Bento, A. P., Chambers, J., Mendez, D., Mutowo, P., Atkinson, F., Bellis, L. J., Cibrián-Uhalte, E., and et al. (2017). The chembl database in 2017. *Nucleic Acids Research*, 45(D1):D945–D954.

Gawande, A. (2018). Why doctors hate their computers. *The New Yorker*, 12.

Gene Ontology Consortium (2021). The gene ontology resource: enriching a gold mine. *Nucleic acids research*, 49(D1):D325–D334.

Gettys, C. F. and Fisher, S. (1979). Hypothesis plausibility and hypothesis generation. *Organizational Behavior and Human Performance*, 24(1):93–110.

Gibson, P. G., Qin, L., and Puah, S. H. (2020). Covid-19 acute respiratory distress syndrome (ards): clinical features and differences from typical pre-covid-19 ards. *The Medical journal of Australia*, 213(2):54–56.e1.

Goldberg, S. I., Niemierko, A., and Turchin, A. (2008). Analysis of data errors in clinical research databases. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, page 242–246.

Gonen, H. and Goldberg, Y. (2019). Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *arXiv preprint arXiv:1903.03862*.

Goodman, S. B., Konttinen, Y. T., and Takagi, M. (2014). Joint replacement surgery and the innate immune system. *Journal of long-term effects of medical implants*, 24(4):253–257.

Google (2021a). About knowledge panels.

Google (2021b). How google's knowledge graph works.

Govindan, R., Kratzke, R. A., Herndon, J. E., Niehans, G. A., Vollmer, R., Watson, D., Green, M. R., and Kindler, H. L. (2005). Gefitinib in patients with malignant mesothelioma: a phase ii study by the cancer and leukemia group b. *Clinical Cancer Research*, 11(6):2300–2304.

Goyal, P. and Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94.

Greenhalgh, T. (2002). Intuition and evidence–uneasy bedfellows? *British Journal of General Practice*, 52(478):395–400.

Grover, A. and Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-Augu:855–864.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 2017-Decem:1025–1035.

Harding, S. (1990). The human pharmacology of fluticasone propionate. *Respiratory medicine*, 84:25–29.

Harris, S., Seaborne, A., and Prud'hommeaux, E. (2013). Sparql 1.1 query language. *W3C recommendation*, 21(10):778.

Hastings, J., Owen, G., Dekker, A., Ennis, M., Kale, N., Muthukrishnan, V., Turner, S., Swainston, N., Mendes, P., and Steinbeck, C. (2016). Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic Acids Research*, 44(D1):D1214–D1219.

Hermawan, H., Ramdan, D., and P. Djuansjah, J. R. (2011). Metals for biomedical applications. In *Biomedical Engineering - From Theory to Applications*, volume 1, pages 411–430.

Himmelstein, D. S. and Baranzini, S. E. (2015). Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS computational biology*, 11(7):e1004259.

Himmelstein, D. S., Lizee, A., Hessler, C., Brueggeman, L., Chen, S. L., Hadley, D., Green, A., Khankhanian, P., and Baranzini, S. E. (2017). Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Montreal, QC, Canada. IEEE.

Hogan, A. (2020). The semantic web: Two decades on. *Semantic Web*, 11(1):169–185.

Hogan, A., Blomqvist, E., Cochez, M., D'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J. F., Staab, S., and Zimmermann, A. (2021). *Knowledge Graphs*. Morgan & Claypool.

Hou, P.-Y., Korn, D. R., Melo-Filho, C. C., Wright, D. R., Tropsha, A., and Chirkova, R. (2022). Compact walks: Taming knowledge-graph embeddings with domain- and task-specific pathways.

Howe, K. L., Achuthan, P., Allen, J., Allen, J., Alvarez-Jarreta, J., Ridwan Amode, M., Armean, I. M., Azov, A. G., Bennett, R., Bhai, J., and et al. (2021). Ensembl 2021. *Nucleic Acids Research*, 49(D1):D884–D891.

Huang, P., Huang, Y., Wang, W., and Wang, L. (2014). Deep embedding network for clustering. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pages 1532–1537. IEEE Computer Society.

Hubauer, T., Lamparter, S., Haase, P., and Herzig, D. (2018). Use cases of the industrial knowledge graph at siemens. *CEUR Workshop Proceedings*, 2180.

Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., and et al. (2002). The ensembl genome database project. *Nucleic Acids Research*, 30(1):38–41.

Hughes, J. P., Rees, S., Kalindjian, S. B., and Philpott, K. L. (2011). Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–49.

Hunter, L. E. (2017). Knowledge-based biomedical data science. *EPJ Data Science*, 1(1–2):19–25.

Hutchinson, M. (2007). Natalizumab: A new treatment for relapsing remitting multiple sclerosis. *Therapeutics and clinical risk management*, 3(2):259–268.

Ioannidis, V. N., Song, X., Manchanda, S., Li, M., Pan, X., Zheng, D., Ning, X., Zeng, X., and Karypis, G. (2020). Drkg - drug repurposing knowledge graph for covid-19.

Jahnen-Dechent, W., Schäfer, C., Heiss, A., and Grötzinger, J. (2001). Systemic inhibition of spontaneous calcification by the serum protein α 2-hs glycoprotein/fetuin. *Zeitschrift für Kardiologie*, 90(3):47–56.

Jain, N., Kalo, J.-C., Balke, W.-T., and Krestel, R. (2021). Do embeddings actually capture knowledge graph semantics? In Verborgh, R., Hose, K., Paulheim, H., Champin, P.-A., Maleshkova, M., Corcho, O., Ristoski, P., and Alam, M., editors, *The Semantic Web*, pages 143–159, Cham. Springer International Publishing.

Jeffery, P. G. and Hutchison, D. (1981). *CHAPTER 18 - Cobalt*, page 155–159. Pergamon Series in Analytical Chemistry. Butterworth-Heinemann, third edit edition.

Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, 1:687–696.

Ji, G., Liu, K., He, S., and Zhao, J. (2016). Knowledge graph completion with adaptive sparse transfer matrix. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, page 985–991.

Jiang, J., Chen, J., Gu, T., Choo, K. K. R., Liu, C., Yu, M., Huang, W., and Mohapatra, P. (2019). Anomaly detection with graph convolutional networks for insider threat and fraud detection. *Proceedings - IEEE Military Communications Conference MILCOM*, 2019-Novem.

Jiang, J. Y., Li, Z., Ju, C. J., and Wang, W. (2020). Maru: Meta-context aware random walks for heterogeneous network representation learning. *International Conference on Information and Knowledge Management, Proceedings*, page 575–584.

Kanehisa, M., Goto, S., Kawashima, S., and Nakaya, A. (2002). The kegg databases at genomenet. *Nucleic Acids Research*, 30(1):42–46.

Kiesling, E., Ekelhart, A., Kurniawan, K., and Ekaputra, F. (2019). The sepses knowledge graph: An integrated resource for cybersecurity. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11779 LNCS:198–214.

Kilicoglu, H., Shin, D., Fiszman, M., Rosemblat, G., and Rindflesch, T. C. (2012). SemMedDB: a PubMed-scale repository of biomedical semantic predications. *Bioinformatics*, 28(23):3158–3160.

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., and et al. (2021). Pubchem in 2021: New data content and improved web interfaces. *Nucleic Acids Research*, 49(D1):D1388–D1395.

Kinney, M. A., Vo, L. T., Frame, J. M., Barragan, J., Conway, A. J., Li, S., Wong, K.-K., Collins, J. J., Cahan, P., North, T. E., and et al. (2019). A systems biology pipeline identifies regulatory networks for stem cell engineering. *Nature Biotechnology*, 37(7):810–818.

Kinsman, L., Rotter, T., James, E., Snow, P., and Willis, J. (2010). What is a clinical pathway? development of a definition to inform the debate. *BMC medicine*, 8:31.

Kishore, B. K., Gejyo, F., and Arakawa, M. (1983). Alpha 2hs-glycoprotein in the serum and urine of patients with renal diseases. *Postgraduate medical journal*, 59(691):304–307.

Kleinberg, J. M. (1999). Hubs, authorities, and communities. *ACM computing surveys (CSUR)*, 31(4es):5–es.

Kong, X., Yu, P. S., Ding, Y., and Wild, D. J. (2012). Meta path-based collective classification in heterogeneous information networks. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 1567–1571, New York, NY, USA. Association for Computing Machinery.

Korn, D., Bobrowski, T., Li, M., Kebede, Y., Wang, P., Owen, P., Vaidya, G., Muratov, E., Chirkova, R., Bizon, C., and Tropsha, A. (2020). COVID-KOP: integrating emerging COVID-19 data with the ROBOKOP database. *Bioinformatics*, 37(4):586–587.

Korn, D., Thieme, A. J., Alves, V. M., Yeakey, M., Borba, J. V., Capuzzi, S. J., Fecho, K., Bizon, C., Edwards, S. W., Chirkova, R., Colvis, C. M., Southall, N. T., Austin, C. P., Muratov, E. N., and Tropsha, A. (2022). Defining clinical outcome pathways. *Drug Discovery Today*, 27(6):1671–1678.

Kuhn, D. (1989). Children and adults as intuitive scientists. *Psychological review*, 96(4):674–689.

Landhuis, E. (2016). Scientific literature: Information overload. *Nature*, 535(7612):457–458.

Landrum, M. J., Lee, J. M., Riley, G. R., Jang, W., Rubinstein, W. S., Church, D. M., and Maglott, D. R. (2014). Clinvar: public archive of relationships among sequence variation and human phenotype. *Nucleic Acids Research*, 42(D1):D980–D985.

Lao, N., Mitchell, T., and Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, page 529–539, USA. Association for Computational Linguistics.

Lea, A. P. and McTavish, D. (1997). Atorvastatin. a review of its pharmacology and therapeutic potential in the management of hyperlipidaemias. *Drugs*, 53(5):828–847.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Lee, J. L., Kim, J. Y., Ryu, M. H., Kang, H. J., Chang, H. M., Kim, T. W., Lee, H., Park, J. H., Kim, H. C., Kim, J. S., and et al. (2006). Response to imatinib in kit- and pdgfra-wild type gastrointestinal stromal associated with neurofibromatosis type 1. *Digestive Diseases and Sciences*, 51(6):1043–1046.

Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A., and Peysakhovich, A. (2019). Pytorch-biggraph: A large-scale graph embedding system. *arXiv*.

Li, Q., Cheng, T., Wang, Y., and Bryant, S. H. (2010). Pubchem as a public resource for drug discovery. *Drug Discovery Today*, 15(23–24):1052–1057.

Li, Y., Shi, C., Yu, P. S., and Chen, Q. (2014). Hrank: A path based ranking method in heterogeneous information network. In Li, F., Li, G., Hwang, S.-w., Yao, B., and Zhang, Z., editors, *Web-Age Information Management*, pages 553–565, Cham. Springer International Publishing.

Liang, J., Ajwani, D., Nicholson, P. K., Sala, A., and Parthasarathy, S. (2016). What links alice and bob? matching and ranking semantic patterns in heterogeneous networks. In *25th International World Wide Web Conference, WWW 2016*, pages 879–889, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Likas, A., Vlassis, N., and J. Verbeek, J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461.

Lim, K. H. and Datta, A. (2012). Finding twitter communities with common interests using following links of celebrities. In *Proceedings of the 3rd International Workshop on Modeling Social Media*, MSM '12, page 25–32, New York, NY, USA. Association for Computing Machinery.

Lin, H., Liu, Y., Wang, W., Yue, Y., and Lin, Z. (2017). Learning entity and relation embeddings for knowledge resolution. *Procedia Computer Science*, 108:345–354.

Lipscomb, C. E. (2000). Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265–266.

Liu, J., Ma, S., Ji, Q., Cao, C., Han, Y., Wang, X., and Zhang, W. (2021). The role of pulmonary mast cells activation and degranulation in the process of increased pulmonary artery pressure. *General physiology and biophysics*, 40(3):183–195.

Longhi, A. and Rizzoli, I. O. (2021). Metformin as maintenance therapy in patients with bone sarcoma and high risk of relapse.

Lonsdale, J., Thomas, J., Salvatore, M., Phillips, R., Lo, E., Shad, S., Hasz, R., Walters, G., Garcia, F., Young, N., and et al. (2013). The genotype-tissue expression (gtex) project. *Nature Genetics*, 45(6):580–585.

Lopes, L. F. and Bacchi, C. E. (2010). Imatinib treatment for gastrointestinal stromal tumour (gist). *Journal of cellular and molecular medicine*, 14(1–2):42–50.

Lu, Z. (2011). Pubmed and beyond: A survey of web tools for searching biomedical literature. *Database*, 2011:baq036.

Lu Wang, L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R., Liu, Z., Merrill, W., and et al. (2020). Cord-19: The covid-19 open research dataset. *arXiv*.

Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., Peng, J., Chen, L., and Zeng, J. (2017). A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nature communications*, 8(1):1–13.

Luscombe, N. M., Greenbaum, D., and Gerstein, M. (2001). What is bioinformatics? a proposed definition and overview of the field. *Methods of Information in Medicine*, 40(4):346–358.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 5.1, pages 281–297. Oakland, CA, USA.

Malone, R. W., Tisdall, P., Fremont-Smith, P., Liu, Y., Huang, X.-P. P., White, K. M., Miorin, L., Olmo, E. M. D., Alon, A., Delaforge, E., and et al. (2021). Covid-19: Famotidine, histamine, mast cells, and mechanisms. *Frontiers in Pharmacology*, 12:rs.3.rs–30934/v2.

Malyshev, S., Krötzsch, M., González, L., Gonsior, J., and Bielefeldt, A. (2018). Getting the most out of wikidata: Semantic technology usage in wikipedia's knowledge graph. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11137 LNCS:376–394.

Marie, P. J., Fromigué, O., and Modrowski, D. (2015). *Chapter 4 - Deregulation of osteoblast differentiation in primary bone cancers*, page 39–54. Academic Press.

Mattingly, C. J., Colby, G. T., Forrest, J. N., and Boyer, J. L. (2003). The comparative toxicogenomics database (ctd). *Environmental health perspectives*, 111(6):793–795.

Mazumder, S. and Liu, B. (2017). Context-aware path ranking for knowledge base completion. In Sierra, C., editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 1195–1201, Melbourne, Australia. International Joint Conferences on Artificial Intelligence Organization.

McDaid, A. F., Greene, D., and Hurley, N. (2011). Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*.

McMurry, J. A., Köhler, S., Washington, N. L., Balhoff, J. P., Borromeo, C., Brush, M., Carbon, S., Conlin, T., Dunn, N., Engelstad, M., and et al. (2016). Navigating the phenotype frontier: The monarch initiative. *Genetics*, 203(4):1491–1495.

Mehta, D., Jackson, R., Paul, G., Shi, J., and Sabbagh, M. (2017). Why do trials for alzheimer's disease drugs keep failing? a discontinued drug perspective for 2010-2015. *Expert opinion on investigational drugs*, 26(6):735–739.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., and et al. (2012). Chembl: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., and et al. (2019). Chembl: Towards direct deposition of bioassay data. *Nucleic Acids Research*.

Mestres, J., Gregori-Puigjané, E., Valverde, S., and Solé, R. V. (2008). Data completeness—the achilles heel of drug-target networks. *Nature Biotechnology*, 26(9):983–984.

Mettang, T. and Kremer, A. E. (2015). Uremic pruritus. *Kidney international*, 87(4):685–691.

Mi, H., Ebert, D., Muruganujan, A., Mills, C., Albou, L. P., Mushayamaha, T., and Thomas, P. D. (2021). Panther version 16: A revised family classification, tree-based classification tool, enhancer regions and extensive api. *Nucleic Acids Research*, 49(D1):D394–D403.

Might, M. (2012). Hunting down my son's killer.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, pages 1–12, Scottsdale, Arizona. ICLR.

Molenaar, R. J., Coelen, R. J., Khurshed, M., Roos, E., Caan, M. W., Van Linde, M. E., Kouwenhoven, M., Bramer, J. A., Bovée, J. V., Mathôt, R. A., and et al. (2017). Study protocol of a phase ib/ii clinical trial of metformin and chloroquine in patients with idh1-mutated or idh2-mutated solid tumours. *BMJ Open*, 7(6).

Moore, E. F. (1959). The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press.

Morgan, M. M., Johnson, B. P., Livingston, M. K., Schuler, L. A., Alarid, E. T., Sung, K. E., and Beebe, D. J. (2016). Personalized in vitro cancer models to predict therapeutic response: Challenges and a framework for improvement. *Pharmacology and Therapeutics*, 165:79–92.

Morton, K., Wang, P., Bizon, C., Cox, S. S., Balhoff, J., Kebede, Y., Fecho, K., and Tropsha, A. (2019). Robokop: An abstraction layer and user interface for knowledge graphs to support question answering. *Bioinformatics*, 35(24):5382–5384.

Moser, H. W., Raymond, G. V., Lu, S.-E., Muenz, L. R., Moser, A. B., Xu, J., Jones, R. O., Loes, D. J., Melhem, E. R., Dubey, P., and et al. (2005). Follow-up of 89 asymptomatic patients with adrenoleukodystrophy treated with lorenzo's oil. *Archives of neurology*, 62(7):1073–80.

Mousavizadeh, L. and Ghasemi, S. (2021). Genotype and phenotype of covid-19: Their roles in pathogenesis. *Journal of Microbiology, Immunology and Infection*, 54(2):159–163.

Mungall, C. J., McMurry, J. A., Kohler, S., Balhoff, J. P., Borromeo, C., Brush, M., Carbon, S., Conlin, T., Dunn, N., Engelstad, M., and et al. (2017). The monarch initiative: An integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Research*, 45(D1):D712–D722.

Murray, J. F., Schraufnagel, D. E., and Hopewell, P. C. (2015). Treatment of tuberculosis: A historical perspective. *Annals of the American Thoracic Society*, 12(12):1749–1759.

National Center for Advancing Translational Sciences (2021). Translator reasoner api version 1.2.0.

National Center for Biotechnology Information (US) (1998). *Genes and Disease [Internet]*, chapter Adrenoleuk. National Center for Biotechnology Information (US).

Nelson, C. A., Butte, A. J., and Baranzini, S. E. (2019). Integrating biomedical research and electronic health records to create knowledge-based biologically meaningful machine-readable embeddings. *Nature Communications*, 10(1):3045.

Nelson, R. H. (2013). Hyperlipidemia as a risk factor for cardiovascular disease. *Primary Care: Clinics in Office Practice*, 40(1):195–211.

Newman, M. (2010). *Networks: An Introduction*. Oxford University Press, Inc.

Nguengang Wakap, S., Lambert, D. M., Olry, A., Rodwell, C., Gueydan, C., Lanneau, V., Murphy, D., Le Cam, Y., and Rath, A. (2020). Estimating cumulative point prevalence of rare diseases: analysis of the orphanet database. *European Journal of Human Genetics*, 28(2):165–173.

Nguyen, D. T., Mathias, S., Bologa, C., Brunak, S., Fernandez, N., Gaulton, A., Hersey, A., Holmes, J., Jensen, L. J., Karlsson, A., and et al. (2017). Pharos: Collating protein information to shed light on the druggable genome. *Nucleic Acids Research*, 45(D1):D995–D1002.

Nicholson, D. N. and Greene, C. S. (2020). Constructing knowledge graphs and their biomedical applications. *Computational and Structural Biotechnology Journal*, 18:1414–1428.

Nissim, M., van Noord, R., and van der Goot, R. (2020). Fair is better than sensational: Man is to doctor as woman is to doctor. *Computational Linguistics*, 46(2):487–497.

Odone, A. and Odone, M. (1989). Lorenzo's oil: a new treatment for adrenoleukodystrophy. *J Pediatr Neurosci*, 5(1):55–61.

of Health, U. D., Services, H., and Program, N. T. (2021). Adverse outcome pathways.

of Health: Office of Data Science Strategy, N. I. (2021). Open-access data and computational resources to address covid-19.

Oldman, D. and Tanase, D. (2018). Reshaping the knowledge graph by connecting researchers, data and practices in researchspace. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11137 LNCS:325–340.

Oliveira, C. A., Candelária, I. S., Oliveira, P. B., Figueiredo, A., and Caseiro-Alves, F. (2015). Metallosis: A diagnosis not only in patients with metal-on-metal prostheses. *European Journal of Radiology Open*, 2:3–6.

Olson, N. D., Wagner, J., McDaniel, J., Stephens, S. H., Westreich, S. T., Prasanna, A. G., Johanson, E., Boja, E., Maier, E. J., Serang, O., Jáspez, D., Lorenzo-Salazar, J. M., Muñoz-Barrera, A., Rubio-Rodríguez, L. A., Flores, C., Kyriakidis, K., Malousi, A., Shafin, K., Pesout, T., Jain, M., Paten, B., Chang, P.-C., Kolesnikov, A., Nattestad, M., Baid, G., Goel, S., Yang, H., Carroll, A., Eveleigh, R., Bourgey, M., Bourque, G., Li, G., Ma, C., Tang, L., Du, Y., Zhang, S., Morata, J., Tonda, R., Parra, G., Trotta, J.-R., Brueffer, C., Demirkaya-Budak, S., Kabakci-Zorlu, D., Turgut, D., Özem Kalay, Budak, G., Narcı, K., Arslan, E., Brown, R., Johnson, I. J., Dolgoborodov, A., Semenyuk, V., Jain, A., Tetikol, H. S., Jain, V., Ruehle, M., Lajoie, B., Roddey, C., Catreux, S., Mehio, R., Ahsan, M. U., Liu, Q., Wang, K., Ebrahim Sahraeian, S. M., Fang, L. T., Mohiyuddin, M., Hung, C., Jain, C., Feng, H., Li, Z., Chen, L., Sedlazeck, F. J., and Zook, J. M. (2022). Precisionfda truth challenge v2: Calling variants from short and long reads in difficult-to-map regions. *Cell Genomics*, 2(5):100129.

Oprea, T. and Mestres, J. (2012). Drug repurposing: far beyond new targets for old drugs. *The AAPS journal*, 14(4):759–763.

Oprea, T. I., Tropsha, A., Faulon, J.-L. L., and Rintoul, M. D. (2007). Systems chemical biology. *Nature chemical biology*, 3(8):447–450.

Organisation for Economic Cooperation and Development (2020). Oecd series on adverse outcome pathways.

Oughtred, R., Rust, J., Chang, C., Breitkreutz, B. J., Stark, C., Willems, A., Boucher, L., Leung, G., Kolas, N., Zhang, F., and et al. (2021). The biogrid database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions. *Protein Science*, 30(1):187–200.

Overington, J. P., Al-Lazikani, B., and Hopkins, A. L. (2006). How many drug targets are there? *Nature reviews Drug discovery*, 5(12):993–996.

Oyama, O., Paltoo, C., and Greengold, J. (2007). Somatoform disorders. *American Family Physician*, 76(9):1333–1338.

Parvathaneni, V., Kulkarni, N. S., Muth, A., and Gupta, V. (2019). Drug repurposing: a promising tool to accelerate the drug discovery process. *Drug Discovery Today*, 24(10):2076–2085.

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8:489–508. 3.

Pawliczek, P., Patel, R. Y., Ashmore, L. R., Jackson, A. R., Bizon, C., Nelson, T., Powell, B., Freimuth, R. R., Strande, N., Shah, N., and et al. (2018). Clingen allele registry links information about genetic variants. *Human Mutation*, 39(11):1690–1701.

Peake, M. D. (2009). Pemetrexed in the treatment of malignant pleural mesothelioma. *Clinical Practice*, 6(4):569.

Pedersen, M. F., Wróbel, T. M., Märcher-Rørsted, E., Pedersen, D. S., Møller, T. C., Gabriele, F., Pedersen, H., Matosiuk, D., Foster, S. R., Bouvier, M., and et al. (2020). Biased agonism of clinically approved μ-opioid receptor agonists and trv130 is not controlled by binding and signaling kinetics. *Neuropharmacology*, 166:107718.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 701–710.

Polamreddy, P. and Gattu, N. (2019). The drug repurposing landscape from 2012 to 2017: evolution, challenges, and possible solutions. *Drug Discovery Today*, 24(3):789–795.

Povey, S., Lovering, R., Bruford, E., Wright, M., Lush, M., and Wain, H. (2001). The hugo gene nomenclature committee (hgnc). *Human Genetics*, 109(6):678–680.

Pryor, R. and Cabreiro, F. (2015). Repurposing metformin: An old drug with new tricks in its binding pockets. *Biochemical Journal*, 471(3):307–322.

Pushpakom, S., Iorio, F., Eyers, P. A., Escott, K. J., Hopper, S., Wells, A., Doig, A., Guilliams, T., Latimer, J., McNamee, C., and et al. (2019). Drug repurposing: Progress, challenges and recommendations. *Nature Reviews Drug Discovery*, 18(1):41–58.

Putignano, V., Rosato, A., Banci, L., and Andreini, C. (2018). Metalpdb in 2018: A database of metal sites in biological macromolecular structures. *Nucleic Acids Research*, 46(D1):D459–D464.

Radev, D. R., Qi, H., Wu, H., and Fan, W. (2002). Evaluating web-based question answering systems. *Proceedings of the 3rd International Conference on Language Resources and Evaluation, LREC 2002*, pages 1153–1156.

Rena, G., Hardie, D. G., and Pearson, E. R. (2017). The mechanisms of action of metformin. *Diabetologia*, 60(9):1577–1585.

Resnik, D. B. (2001). Setting biomedical research priorities: justice, science, and public participation. *Kennedy Institute of Ethics Journal*, 11(2):181–204.

Reuber, M., House, A. O., Pukrop, R., Bauer, J., and Elger, C. E. (2003). Somatization, dissociation and general psychopathology in patients with psychogenic non-epileptic seizures. *Epilepsy research*, 57(2-3):159–167.

Rice, J. A. (2006). *Mathematical statistics and data analysis*. Cengage Learning, Boston, Massachusetts.

Richardson, P., Griffin, I., Tucker, C., Smith, D., Oechsle, O., Phelan, A., and Stebbing, J. (2020). Baricitinib as potential treatment for 2019-ncov acute respiratory disease. *Lancet*, 395(10223):e30.

Risch, T. (2009). *Distributed Architecture*, page 879–879. Springer US.

Robinson, I., Webber, J., and Eifrem, E. (2015). *Graph Databases: New Opportunities for Connected Data*. " O'Reilly Media, Inc.".

Robinson, P. N., Köhler, S., Bauer, S., Seelow, D., Horn, D., and Mundlos, S. (2008). The human phenotype ontology: A tool for annotating and analyzing human hereditary disease. *The American Journal of Human Genetics*.

Rodriguez, M. A. (2015). The gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages*, DBPL 2015, page 1–10, New York, NY, USA. Association for Computing Machinery.

Rodriguez-Esteban, R. (2022). The speed of information propagation in the scientific network distorts biomedical research. *PeerJ*, 10:e12764.

Röhl, J. (2012). Mechanisms in biomedical ontology. *Journal of Biomedical Semantics*, 3(2):1–14.

Romesburg, J. W., Wasserman, P. L., and Schoppe, C. H. (2010). Metallosis and metal-induced synovitis following total knee arthroplasty: Review of radiographic and ct findings. *Journal of Radiology Case Reports*, 4(9):7–17.

Ronfeldt, D. and Arquilla, J. (2020). Whose story wins: Rise of the noosphere, noopolitik, and information-age statecraft. Technical report, RAND Corporation.

Roth, B. L., Sheffer, D. J., and Kroeze, W. K. (2004). Magic shotguns versus magic bullets: Selectively non-selective drugs for mood disorders and schizophrenia. *Nature Reviews Drug Discovery*, 3(4):353–359.

Rusiecki, J. A., De Roos, A., Lee, W. J., Dosemeci, M., Lubin, J. H., Hoppin, J. A., Blair, A., and Alavanja, M. C. (2004). Cancer incidence among pesticide applicators exposed to atrazine in the agricultural health study. *Journal of the National Cancer Institute*, 96(18):1375–1382.

Sabesin, S. M., Berlin, R. G., Humphries, T. J., Bradstreet, D. C., Walton-Bowen, K. L., and Zaidi, S. (1991). Famotidine relieves symptoms of gastroesophageal reflux disease and heals erosions and ulcerations: Results of a multicenter, placebo-controlled, dose-ranging study. *Archives of Internal Medicine*, 151(12):2394–2400.

Sadowski, G. and Rathle, P. (2014). Fraud detection: Discovering connections with graph databases. *White Paper-Neo Technology-Graphs are Everywhere*, 13:17.

Sam T., R. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(22):2323–2326.

Sand, I. K. (2018). The role of diet in multiple sclerosis: mechanistic connections and current evidence. *Current nutrition reports*, 7(3):150–160.

Sanders, R. (1987). The Pareto Principle: Its Use And Abuse. *Journal of Services Marketing*, 1(2):37–40.

Schett, G., Sticherling, M., and Neurath, M. F. (2020). Covid-19: risk for cytokine targeting in chronic inflammatory diseases? *Nature Reviews Immunology*, 20(5):271–272.

Schreurs, A. S., Torres, S., Truong, T., Moyer, E. L., Kumar, A., Tahimic, C. G., Alwood, J. S., and Globus, R. K. (2020). Skeletal tissue regulation by catalase overexpression in mitochondria. *American Journal of Physiology - Cell Physiology*, 319(4):C734–C745.

SciBite (2021). Scibite — about us.

Shefchek, K. A., Harris, N. L., Gargano, M., Matentzoglu, N., Unni, D., Brush, M., Keith, D., Conlin, T., Vasilevsky, N., Zhang, X. A., and et al. (2020). The monarch initiative in 2019: An integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Research*, 48(D1):D704–D715.

Shi, B. and Weninger, T. (2016). Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, page 101–102, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Shi, C., Li, Y., Zhang, J., Sun, Y., and Yu, P. S. (2017). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37.

Sibson, R. (1973). Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34.

Singh, T. U., Parida, S., Lingaraju, M. C., Kesavan, M., Kumar, D., and Singh, R. K. (2020). Drug repurposing approach to fight COVID-19. *Pharmacological Reports*, 72(6):1479–1508.

Smith, C. M., Hayamizu, T. F., Finger, J. H., Bello, S. M., McCright, I. J., Xu, J., Baldarelli, R. M., Beal, J. S., Campbell, J., Corbani, L. E., and et al. (2019). The mouse gene expression database (gxd): 2019 update. *Nucleic Acids Research*, 47(D1):D774–D779.

Snead III, O. C. and Gibson, K. M. (2005). Gamma-hydroxybutyric acid. *New England Journal of Medicine*, 352(26):2721–2732.

Society for Advancement of AOPs (2021). Aop-wiki.

Sosa, D. N., Derry, A., Guo, M., Wei, E., Brinton, C., and Altman, R. B. (2019). A literature-based knowledge graph embedding method for identifying drug repurposing opportunities in rare diseases. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2020*, pages 463–474, Bethesda, Maryland. World Scientific, ISCB.

Sosa, D. N., Derry, A., Guo, M., Wei, E., Brinton, C., and Altman, R. B. (2020). A literature-based knowledge graph embedding method for identifying drug repurposing opportunities in rare diseases. *Pacific Symposium on Biocomputing*, 25(2020):463–474.

Spangler, S., Wilkins, A. D., Bachman, B. J., Nagarajan, M., Dayaram, T., Haas, P., Regenbogen, S., Pickering, C. R., Comer, A., Myers, J. N., and et al. (2014). Automated hypothesis generation based on mining scientific literature. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 1877–1886.

Square, R. (2020). Covid-19: Famotidine, histamine, mast cells, and mechanisms. *Preprint*.

Stark, C., Breitkreutz, B. J., Reguly, T., Boucher, L., Breitkreutz, A., and Tyers, M. (2006). Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(Database issue).

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A core of semantic knowledge. In *16th International World Wide Web Conference, WWW2007*, page 697–706.

Sun, Y. and Han, J. (2013). Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2):20–28.

Swanson, D. R. (1986). Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in Biology and Medicine*.

Sybrandt, J., Tyagin, I., Shtutman, M., and Safro, I. (2020). Agatha: Automatic graph-mining and transformer based hypothesis generation approach. *arXiv*.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160.

Team, T. N. (2022). The neo4j cypher manual v4.4.

Thafar, M. A., Olayan, R. S., Ashoor, H., Albaradei, S., Bajic, V. B., Gao, X., Gojobori, T., and Essack, M. (2020). Dtigems+: drug–target interaction prediction using graph embedding, graph mining, and similarity-based techniques. *Journal of Cheminformatics*, 12(1):44.

Thambisetty, M. and Aging, N. I. O. (2021). Novel approach to treat alzheimer's disease: Dream study perspective.

Timbrell, J. (2001). *Introduction to toxicology*. CRC Press.

Tollefsen, K. E., Scholz, S., Cronin, M. T., Edwards, S. W., de Knecht, J., Crofton, K., Garcia-Reyero, N., Hartung, T., Worth, A., and Patlewicz, G. (2014). Applying Adverse Outcome Pathways (AOPs) to support Integrated Approaches to Testing and Assessment (IATA). *Regulatory Toxicology and Pharmacology*, 70(3):629–640.

Tran, J. B. and Krężel, A. (2021). InterMetalDB: A Database and Browser of Intermolecular Metal Binding Sites in Macromolecules with Structural Information. *Journal of Proteome Research*, 20(4):1889–1901.

Tweedie, S., Braschi, B., Gray, K., Jones, T. E., Seal, R. L., Yates, B., and Bruford, E. A. (2021). Genenames.org: The hgnc and vgnc resources in 2021. *Nucleic Acids Research*, 49(D1):D939–D946.

University, J. H. and Medicine (2021). Covid-19 map - johns hopkins coronavirus resource center.

Ursu, O., Holmes, J., Knockel, J., Bologa, C. G., Yang, J. J., Mathias, S. L., Nelson, S. J., and Oprea, T. I. (2016). Drugcentral: online drug compendium. *Nucleic acids research*, 45(D1):D932–D939.

Ursu, O., Holmes, J., Knockel, J., Bologa, C. G., Yang, J. J., Mathias, S. L., Nelson, S. J., and Oprea, T. I. (2017). Drugcentral: online drug compendium. *Nucleic Acids Research*, 45(D1):D932–D939.

Uschold, M. and King, M. (1995). Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*.

Valdez, R., Ouyang, L., and Bolen, J. (2016). Public health and rare diseases: Oxymoron no more. *Preventing Chronic Disease*, 13:150491.

Van den Broeck, W. M. (2015). Chapter 3 - drug targets, target identification, validation, and screening. In Wermuth, C. G., Aldous, D., Raboisson, P., and Rognan, D., editors, *The Practice of Medicinal Chemistry (Fourth Edition)*, pages 45–70. Academic Press, San Diego, fourth edition edition.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11):2579–2605.

Vazakidou, M. E., Magkouta, S., Moschos, C., Psallidas, I., Pappas, A., Psarra, K., and Kalomenidis, I. (2015). Temsirolimus targets multiple hallmarks of cancer to impede mesothelioma growth in vivo. *Respirology*, 20(8):1263–1271.

Čebirić, v., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., and Zneika, M. (2019). Summarizing semantic graphs: A survey. *The VLDB Journal*, 28(3):295–327.

Visentin, M., Zhao, R., and Goldman, I. D. (2012). The antifolates. *Hematology/Oncology Clinics*, 26(3):629–648.

Vogt, L. (2021). FAIR data representation in times of eScience: a comparison of instance-based and class-based semantic representations of empirical data using phenotype descriptions as example. *Journal of Biomedical Semantics*, 12(1):20.

Waagmeester, A., Stupp, G., Burgstaller-Muehlbacher, S., Good, B. M., Griffith, M., Griffith, O. L., Hanspers, K., Hermjakob, H., Hudson, T. S., Hybiske, K., and et al. (2020). Science forum: Wikidata as a knowledge graph for the life sciences. *Elife*, 9:e52614.

Wang, P. (2021). Strider.

Wang, Q., Li, M., Wang, X., Parulian, N., Han, G., Ma, J., Tu, J., Lin, Y., Zhang, H., Liu, W., et al. (2020). Covid-19 literature knowledge graph construction and drug repurposing report generation.

Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Wang, X., Wang, D., Xu, C., He, X., Cao, Y., and Chua, T.-S. (2019). Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33 of *AAAI'19/IAAI'19/EAAI'19*, pages 5329–5336, Honolulu, Hawaii, USA. AAAI Press.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. *Proceedings of the National Conference on Artificial Intelligence*, 2:1112–1119.

Webber, J. (2012). A programmatic introduction to neo4j. *SPLASH'12 - Proceedings of the 2012 ACM Conference on Systems, Programming, and Applications: Software for Humanity*, page 217.

Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36.

Westenfeld, R., Schäfer, C., Smeets, R., Brandenburg, V. M., Floege, J., Ketteler, M., and Jahnen-Dechent, W. (2007). Fetuin-a (ahsg) prevents extraosseous calcification induced by uraemia and phosphate challenge in mice. *Nephrology Dialysis Transplantation*, 22(6):1537–1546.

Whirl-Carrillo, M., Huddart, R., Gong, L., Sangkuhl, K., Thorn, C. F., Whaley, R., and Klein, T. E. (2021). An evidence-based framework for evaluating pharmacogenomics knowledge for personalized medicine. *Clinical Pharmacology and Therapeutics*, 110(3):563–572.

Whirl-Carrillo, M., McDonagh, E. M., Hebert, J. M., Gong, L., Sangkuhl, K., Thorn, C. F., Altman, R. B., and Klein, T. E. (2012). Pharmacogenomics knowledge for personalized medicine. *Clinical Pharmacology and Therapeutics*, 92(4):414–417.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018.

Williams, A. J., Harland, L., Groth, P., Pettifer, S., Chichester, C., Willighagen, E. L., Evelo, C. T., Blomberg, N., Ecker, G., Goble, C., and Mons, B. (2012). Open phacts: semantic interoperability for drug discovery. *Drug Discovery Today*, 17(21):1188–1198.

Wishart, D. S., Knox, C., Guo, A. C., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., and Hassanali, M. (2008). Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research*, 36(Database issue):D901–6.

Wishart, D. S., Knox, C., Guo, A. C., Eisner, R., Young, N., Gautam, B., Hau, D. D., Psychogios, N., Dong, E., Bouatra, S., and et al. (2009). Hmdb: A knowledgebase for the human metabolome. *Nucleic Acids Research*, 37(SUPPL. 1).

Wishart, D. S., Tzur, D., Knox, C., Eisner, R., Guo, A. C., Young, N., Cheng, D., Jewell, K., Arndt, D., Sawhney, S., and et al. (2007). Hmdb: The human metabolome database. *Nucleic Acids Research*, 35(SUPPL. 1).

Wittwehr, C., Aladjov, H., Ankley, G., Byrne, H. J., de Knecht, J., Heinzle, E., Klambauer, G., Landesmann, B., Luijten, M., MacKay, C., Maxwell, G., Meek, M. E. B., Paini, A., Perkins, E., Sobanski, T., Villeneuve, D., Waters, K. M., and Whelan, M. (2017). How Adverse Outcome Pathways Can Aid the Development and Use of Computational Prediction Models for Regulatory Toxicology. *Toxicological sciences : an official journal of the Society of Toxicology*, 155(2):326–336.

Wood, E. C., Glen, A. K., Kvarfordt, L. G., Womack, F., Acevedo, L., Yoon, T. S., Ma, C., Flores, V., Sinha, M., Chodpathumwan, Y., Termehchy, A., Roach, J. C., Mendoza, L., Hoffman, A. S., Deutsch, E. W., Koslicki, D., and Ramsey, S. A. (2021). RTX-KG2: a system for building a semantically standardized knowledge graph for translational biomedicine.

Woolley, A. and Kostopoulou, O. (2013). Clinical intuition in family medicine: More than first impressions. *Annals of Family Medicine*, 11(1):60–66.

Wu, C., MacLeod, I., and Su, A. I. (2013). Biogps and mygene.info: Organizing online, gene-centric information. *Nucleic Acids Research*, 41(D1).

Wu, C., Su, A., Tsueng, G., Cano, M. A., Xu, C., Yao, Y., Avila, R., Mullen, J., Hu, E., and Haag, E. (2022). About MyChem.info.

Xiao, H., Huang, M., Hao, Y., and Zhu, X. (2015). Transa: An adaptive approach for knowledge graph embedding. *CoRR*, abs/1509.05490.

Xin, J., Mark, A., Afrasiabi, C., Tsueng, G., Juchler, M., Gopal, N., Stupp, G. S., Putman, T. E., Ainscough, B. J., Griffith, O. L., and et al. (2016). High-performance web services for querying gene and variant annotation. *Genome Biology*, 17(1).

Yan, V. K. C., Li, X., Ye, X., Ou, M., Luo, R., Zhang, Q., Tang, B., Cowling, B. J., Hung, I., Siu, C. W., Wong, I. C. K., Cheng, R. C. K., and Chan, E. W. (2021). Drug repurposing for the treatment of covid-19: A knowledge graph approach. *Advanced Therapeutics*, 4(10):2100179.

Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., and Tang, J. (2020). Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 1666–1676, New York, NY, USA. Association for Computing Machinery.

Yin, H., Wang, W., Wang, H., Chen, L., and Zhou, X. (2017). Spatial-aware hierarchical collaborative deep learning for poi recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2537–2551.

Zhang, P., Bu, Y., Jiang, P., Shi, X., Lun, B., Chen, C., Syafiandini, A. F., Ding, Y., and Song, M. (2021). Toward a coronavirus knowledge graph. *Genes*, 12(7):998.

Zheng, W., Zou, L., Peng, W., Yan, X., Song, S., and Zhao, D. (2016). Semantic SPARQL similarity search over RDF knowledge graphs. *Proceedings of the VLDB Endowment*, 9(11):840–851.

Zhou, Y., Wang, F., Tang, J., Nussinov, R., and Cheng, F. (2020). Artificial intelligence in COVID-19 drug repurposing. *The Lancet. Digital health*, 2(12):e667–e676.

Zhu, Y., Che, C., Jin, B., Zhang, N., Su, C., and Wang, F. (2020). Knowledge-driven drug repurposing using a comprehensive drug knowledge graph. *Health Informatics Journal*, 26(4):2737–2750.