HHS Public Access

Author manuscript

Big Data Cogn Comput. Author manuscript; available in PMC 2022 August 04.

Published in final edited form as:

Big Data Cogn Comput. 2022 March ; 6(1): . doi:10.3390/bdcc6010027.

Optimizations for Computing Relatedness in Biomedical Heterogeneous Information Networks: SemNet 2.0

Anna Kirkpatrick[1,2], **Chidozie Onyeze**[1,2], **David Kartchner**[1,3], **Stephen Allegri**[1,4], **Davi Nakajima An**[1,3], **Kevin McCoy**[1,4], **Evie Davalbhakta**[1], **Cassie S. Mitchell**[1,4,5,*]

[1]Laboratory for Pathology Dynamics, Georgia Institute of Technology and Emory University, Atlanta, GA 30332, USA

[2]School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA

[3]School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA

[4]Department of Biomedical Engineering, Georgia Institute of Technology and Emory University, Atlanta, GA 30332, USA

[5]Machine Learning Center at Georgia Tech, Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract

Literature-based discovery (LBD) summarizes information and generates insight from large text corpuses. The SemNet framework utilizes a large heterogeneous information network or "knowledge graph" of nodes and edges to compute relatedness and rank concepts pertinent to a user-specified target. SemNet provides a way to perform multi-factorial and multi-scalar analysis of complex disease etiology and therapeutic identification using the 33+ million articles in PubMed. The present work improves the efficacy and efficiency of LBD for end users by augmenting SemNet to create SemNet 2.0. A custom Python data structure replaced reliance on Neo4j to improve knowledge graph query times by several orders of magnitude. Additionally, two randomized algorithms were built to optimize the HeteSim metric calculation for computing metapath similarity. The unsupervised learning algorithm for rank aggregation (ULARA), which ranks concepts with respect to the user-specified target, was reconstructed using derived mathematical proofs of correctness and probabilistic performance guarantees for optimization. The upgraded ULARA is generalizable to other rank aggregation problems outside of SemNet. In summary, SemNet 2.0 is a comprehensive open-source software for significantly faster, more effective, and user-friendly means of automated biomedical LBD. An example case is performed to rank relationships between Alzheimer's disease and metabolic co-morbidities.

*Correspondence: cassie.mitchell@bme.gatech.edu or csmitch@emory.edu.

Author Contributions: Conceptualization, A.K., D.K. and C.S.M.; methodology, A.K., C.O., D.K. and C.S.M.; software, A.K., D.K., S.A., D.N.A. and K.M.; validation, A.K. and C.O.; formal analysis, A.K., C.O. and S.A.; investigation, A.K., C.O., D.K., S.A., D.N.A., K.M., E.D. and C.S.M.; resources, C.S.M.; data curation, D.K., D.N.A., S.A. and K.M.; writing—original draft preparation, A.K., C.O., S.A, E.D. and C.S.M.; writing—review and editing, A.K., C.O., D.K., S.A., D.N.A., K.M., E.D. and C.S.M.; visualization, D.N.A. and E.D.; supervision, C.S.M.; project administration, C.S.M.; funding acquisition, A.K. and C.S.M. All authors have read and agreed to the published version of the manuscript.

Keywords

HeteSim; ULARA; SemNet; Alzheimer's disease; natural language processing; machine learning; text mining; biomedical knowledge graph; relatedness; rank aggregation

1. Introduction

Biomedical research, like the human body itself, is a complex network of interrelated concepts and relationships that make up a greater whole. There are more than 33 million abstracts and counting in PubMed, one of the largest and most widely used databases and search engines for biomedical research [1]. Many researchers use PubMed, or similar databases, to look up information using specific keywords. However, it is impossible to manually read and synthesize all articles across all related topics. The goal of literature-based discovery (LBD), founded by Dr. Swanson in 1986 [2], is concentrating and concatenating conclusions between disparate sources of information to both improve existing insights as well as generate new insights. The field of LBD attempts to capture knowledge from biomedical text and integrate it in a way that makes discovery of new knowledge possible. In Henry et al. [3], LBD techniques were used to discover lecithin-cholesterol acyltransferase (LCAT) as a proposed therapeutic target for cardiac arrest, a target that was later supported via in vivo studies. Additionally, LBD was used to identify repurposed drugs for the COVID-19 pandemic [4]. While LBD has the potential to be truly transformative, challenges remain to optimize the underlying text mining methodology as well as to make LBD more accessible to domain specialists and clinicians. The presented work optimizes LBD by improving the efficiency and efficacy of LBD in an interactive, open-source Python-based framework called SemNet 2.0.

1.1. Automating the LBD Process

The first step in the LBD process is to model the connections between biomedical concepts in a medium where both humans and computers can easily work with the data. Heterogeneous information networks, or more specifically biomedical concept graphs, provide an exceptional scaffold and starting point for LBD. Modeling biomedical relationships using a graph structure is not ubiquitous in the LBD field, though it is common and wrought with research potential. Various methods, including those described in Cameron et al. [5], Crichton el al. [6], and Sang et al. [7], have used graph-based approaches to perform LBD to great success (all of which vary in how the graphs are constructed and analyzed). Simply put, a biomedical concept graph is the most intuitive and flexible representation available to model semantic predications, especially given the heterogeneous nature of the data and overall direction the LBD field is currently moving. As a brief aside, the terminology used to describe these graphs is borrowed from graph theory and social network analysis; biomedical concepts are referred to as "nodes" and connections between concepts are referred to as "edges". In the context of the present study, this data representation is built as a directed graph in which each node corresponds to a Unified Medical Language System (UMLS) biomedical concept (Alzheimer's disease, insulin, COVID-19, etc.) and each directed edge encodes a UMLS predication (inhibits, treats, causes, etc.) between a source and target. Additionally, each node has an associated

UMLS semantic type (disease or syndrome, gene or genome, therapeutic or preventive procedure, etc.). Each concept–predication–concept relationship within the graph has been extracted from biomedical article abstracts within PubMed via SemRep and stored within the MySQL-based Semantic Medline Database (SemMedDB) [8]. SemMedDB is essentially a table of subject–predicate–object triples, which are manipulated to form an approximately 300,000 node and 20,000,000 edge knowledge graph that combines standardized biomedical concepts and relationships.

The second major step in LBD is using natural language processing and machine learning techniques to identify concepts of interest that are related to a user-specified query. A user, such as a domain specialist, specifies one or more target nodes of interest, which defines the topic upon which the user wishes to discover and ultimately rank related concepts from the literature. The target is analogous to a user entering a keyword to lookup an article in PubMed. Then, using a few other user inputs to constrain the queried biomedical knowledge graph results, such as node type (pharmacologic substance; gene or genome; disease or syndrome, etc.) and/or relation type (treats, affects, inhibits, etc.), the algorithm finds all the related source nodes by examining metapaths in the graph. A metapath is a series of sequential node and relationship types that tie the user-specified target node to the related source nodes identified in the graph.

The third major step in LBD is using machine learning and mathematical optimization to rank the importance of the identified source nodes to the target node. There are various methods of ranking concepts in heterogeneous information networks, often stemming from an assortment of domains [9–11]. The most common methods utilize the graph's metapaths to perform statistical analysis on one more more features of interest. Example features may be simple node or path counts, or they may be more complex, such as the HeteSim metric, which examines metapath similarities [12]. Finally, a ranking algorithm uses the feature scores to provide a final ranking that can provide important context to a domain scientist. That is, of all the identified related concepts of interest, which ones are most important to the user-specified target?

1.2. Overview of SemNet

SemNet was the first Python-based open-source software that enabled all three steps of LBD to be performed using the 33 plus million biomedical abstracts in PubMed [13]. SemNet will be briefly introduced here and is fully described in the original published work by Sedler and Mitchell [13]. First, SemNet had the initial task of building the knowledge graph derived from SemMedDB's semantic relationships and storing it, which it did via an interactive graph database management system called Neo4j. Second, SemNet used py2neo to query the biomedical concept graph constructed in Neo4j to compute metapaths. Third, SemNet adopted a version of the unsupervised learning algorithm for rank aggregation, ULARA, published by Klementiev and colleagues [14] to perform the calculations for feature rankings, including the computation of the HeteSim score. In summary, SemNet enabled a user to easily input a few targets and subsequently retrieve source node importance rankings using machine learning to process millions of biomedical concepts.

1.3. Improving LBD Efficiency and Efficacy with SemNet 2.0

SemNet laid an important foundation for making LBD accessible and usable for domain researchers. However, SemNet simulations were extremely slow, even when performed on high-end servers. The slowness and amount of required computation also limited the amount of detail that could be studied in SemNet. Namely, it limited the length of metapaths that could be ranked (e.g., maximum possible path length for ranking calculation was equal to two); this limit was problematic for a domain researcher wishing to examine more nuanced related concepts that would likely have a longer path length. The present study was largely motivated by the need to enhance SemNet to improve its computation speed, usability and utility. From this point forward, the original SemNet will be referred to as SemNet version 1. The present study performs a full evaluation of speed bottlenecks in SemNet version 1 and proposes and evaluates alternative solutions. The research process to improve speed led to additional mathematical scrutiny of the utilized HeteSim metric and ULARA algorithms in SemNet version 1. Thus, the present study includes both a presentation of optimized mathematical solutions as well as changes to algorithmic and data handling frameworks to increase overall speed. Three major technical improvements were made to create SemNet version 2 (also known as SemNet 2.0): (1) a randomized approximation algorithm for estimating HeteSim scores to improve HeteSim calculation speed; (2) a re-engineered knowledge graph framework that removed reliance on Neo4j to improve metapath and feature computation speed; (3) an improved implementation of the adopted ULARA ranking algorithm.

The first major improvement focused on the efficiency of algorithms utilizing the HeteSim metric. HeteSim-based similarity scoring on heterogeneous information networks has been successfully applied to multiple biomedical research problems [15–20]; therefore, the implementation of a faster HeteSim scoring algorithm will have the potential for significant benefit to the biomedical research community. The main investigative line for algorithm improvements involves approximation algorithms using randomness. An approximation algorithm is a unique algorithm which returns a value within a specified error (generally additive or multiplicative) of the true answer, with some known or bounded probability. The power of approximation algorithms lies in their ability, for some problems, to provide a fast approximation to a solution even when computing the exact solution requires exponential time (assuming ($P \neq NP$)). Though approximation algorithms have existed in the literature for some time, Garey, Graham, and Ullman [21] and Johnson [22] both introduced the idea formally in 1973 and 1974, respectively. Since then, the computer science and combinatorics literature has featured many advancements in the field of randomized approximation algorithms. For an overview of basic techniques and more recent results, see [23–25].

The second major improvement focused on re-engineering the graph data structure to remove query processing bottlenecks and improve overall performance via faster data accessibility. SemNet version 1 uses Neo4j, an efficient graph database management system that employs a specialized declarative query language (Cypher) optimized for graphs, to store and query the biomedical concept graph [26]. At first glance, the choice to use Neo4j is intuitive. It is custom designed to deal with graphs akin to the one SemNet builds, and it has been used before in similar projects to great success [9]. Nonetheless, constantly querying

an externally accessible database to run the HeteSim algorithm, even with the use of multi-threading, proved much slower than desired. This outcome prompted an investigation into alternatives to Neo4j. The evaluated alternative is a locally stored Python nested dictionary graph representation, a data structure that lacks the appealing interfaces of Neo4j but has greatly improved data handling speeds.

The third major improvement focused on the ULARA ranking algorithm. Careful mathematical investigation of ULARA led to the identification of a pertinent flaw in the originally published ULARA algorithm [14]. As noted above, SemNet version 1 had adopted ULARA for aggregating HeteSim scores over multiple metapaths. Fortunately, the specific implementation of ULARA to SemNet version 1 minimized the impact of the identified ULARA flaw on SemNet version 1 results. Nonetheless, a full and precise solution was necessary to correctly fix ULARA and improve the produced rankings. Section 2.1 explains the flaw in the original ULARA [14] and proposes an alternative, which was implemented in SemNet version 2.

The mathematics of the SemNet version 2 improvements are derived in full in subsequent sections. Beyond the mathematics, real-world examples and user studies are used to showcase the improvements and power of SemNet version 2.

1.4. Use Case Example: Alzheimer's Disease and Metabolism

SemNet version 2 was primarily developed for interactive, multi-factorial and multi-scalar relationship exploration in biomedical science and health care. For this study, the primary target node, Alzheimer's disease (AD), was chosen to compare performance of the original SemNet (i.e., SemNet version 1) to the developed SemNet version 2. AD was chosen due to its large degree of connectivity, multi-factorial and heterogeneous nature, and growing relevance in health care—a byproduct of increasing AD deaths and an aging global population [27,28]. AD is traditionally characterized by its tau and amyloid beta protein deposition in neurofibrillary tangles, brain atrophy, and eventual cognitive decline [3]. As researchers delved deeper into the disease, the breadth of risk factors across various domains, such as pharmaceuticals, antecedent disease, psychological profile, and lifestyle, has further increased overall complexity of AD investigation [27,29,30]. This complexity is exacerbated by the difficulty of defining AD sub-populations, a problem that impacts clinical trial patient selection and therapeutic evaluation [31]. Given AD's heterogeneous nature, traditional bioinformatics solutions struggle where the SemNet framework thrives. SemNet version 2 is optimized to work with heterogeneous data, drawing from literature across all biomedical domains to provide concept rankings. The flexibility, efficacy, and efficiency of SemNet version 2 is evaluated using AD as a case study. Thus, Alzheimer's disease (CUI: C0002395) is chosen as the primary target to three diverse sources: insulin (CUI: C0021641), hypothyroidism (CUI: C0020676), and amyloid (CUI: C0002716). Amyloid was chosen as a known "control", where the relationship between amyloid and Alzheimer's disease is well known and validated; therefore, amyloid has many paths and metapaths connecting it to AD [32]. Insulin and hypothyroidism were chosen to assess a newer hypothesis that metabolic syndromes may play a significant role in the onset risk or outcome of AD [31,33]. The nodes of insulin and hypothyroidism have sufficient

connections to AD to be considered relevant but are distant enough, domain wise, to showcase SemNet's flexibility in exploring more nuanced, and lesser cited multi-factorial disease etiology [34,35].

1.5. Definitions and Mathematical Preliminaries

In this section, we will formally define a schema and a knowledge graph/heterogeneous information network. A schema tells us which node and edge types may be present in our knowledge graph, while the knowledge graph tells us which relations apply to specific concepts nodes.

Definition 1. *A schema* $\mathcal{S} = (\mathcal{A}, \mathcal{R})$ *is a set* $\mathcal{A}$ *of node types and a set* $\mathcal{R}$ *of relations. Each relation* $R \in \mathcal{R}$ *has a source type* $A \in \mathcal{A}$ *and a target type* $B \in \mathcal{A}$.

Definition 2. *Let* $S = (\mathcal{A}, \mathcal{R})$ *be a schema with* $|\mathcal{A}| > 1$. *Then, a heterogeneous information network (also called a knowledge graph) is a directed graph* $G = (V, E)$ *with an object type mapping function* $\varphi: V \to \mathcal{A}$ *and a link type mapping function* $\psi: E \to \mathcal{R}$. *If* $e = (u, v) \in E$, *then the source type of* $\psi(e)$ *must be* $\varphi(u)$ *and similarly the target type of* $\psi(e)$ *must be* $\varphi(v)$.

Relations are a key concept in understanding knowledge graphs. We may understand both individual edges and entire metapaths as relations. We start by defining the simplest relation, the self relation.

Definition 3. *The relation I is the self-relation. So,* $a \xrightarrow{I} b$ *if and only if* $a = b$. *We also define the function* δ *by* $\delta(a, b) = 1$ *if* $a = b$ *and* $\delta(a, b) = 0$ *otherwise*.

We now define our primary object of study: the metapath. Note that the metapath may be viewed as a list of node and edge types or as the relation equivalent to the composition of all individual relations in the metapath.

Definition 4. *Let* $\mathcal{S} = (\mathcal{A}, \mathcal{R})$ *be a schema. Then, a metapath* $\mathcal{P}$ *is a sequence of node and edge types, denoted* $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1}$, *with* $A_i \in \mathcal{A}$ *and* $R_i \in \mathcal{R}$. *The length of* $\mathcal{P}$ *is l. Note that a metapath may also be understood as the composition of the relations given by its metaedges:* $R = R_1 \circ R_2 \circ \cdots \circ R_l$. *Let* $p = a_1 a_2 \dots a_{l+1}$ *with* $a_i \in V$ *and* $(a_i, a_{i+1}) \in E$ *be a path in G. Then, p is a path instance of the metapath* $\mathcal{P}$ *if* $\varphi(a_i) = A_i \forall\, i \le l + 1$ *and* $\psi((a_i, a_{i+1})) = R_i \forall\, i \le l$. *We denote the fact that p is a path instance of* $\mathcal{P}$ *by* $p \in \mathcal{P}$.

Given these definitions, we are nearly ready to define the function of interest: *HeteSim*, which was defined by Shi et al. [12]. We start by defining a function *h* which is a non-normalized version of HeteSim.

Definition 5. *Let* $l > 0$. *Let* $\mathcal{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1}$. *Let* $\varphi(s) = A_1$ *and* $\varphi(t) = R_{l+1}$. *Then the non-normalized HeteSim score between s and t with respect to the relevance path* $\mathcal{P}$ *is defined recursively as follows. When* $R_1 \circ R_2 \circ \dots R_l \neq I$,

$$h(s,t \mid R_1 \circ R_2 \circ \cdots \circ R_l) = \frac{1}{|O(s \mid R_1)||I(t \mid R_l)|} \sum_{a \in O(s \mid R_1)} \sum_{b \in I(t \mid R_l)} h(a,b \mid R_2 \circ R_3 \circ \cdots \circ R_l - 1),$$

where $O(s|R_1)$ is the set of out-neighbors of node s based on relation R_1, and $I(t|R_l)$ is the set of in-neighbors of node t based on the relation R_l. In the base case, we define

$$h(a,b \mid I) = \delta(a,b).$$

Note that this definition only works for relevance paths of even length. We will need an extension for paths of odd length.

We briefly explain the definition of HeteSim for odd paths here. For more detail, see Shi et al. [12].

The basic idea to define h for paths of odd length is to transform those paths into paths of even length. Suppose we have a relevance path of odd length $\mathscr{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$. We now modify $\mathscr{P}$ by adding a new object type E and two new relation types R_E and R_F. We then define $\mathscr{P}' = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{\frac{l+1}{2}-1}} A_{\frac{l+1}{2}} \xrightarrow{R_E} E \xrightarrow{R_F} A_{\frac{l+1}{2}+1} \xrightarrow{R_{\frac{l+1}{2}+1}} \cdots \xrightarrow{R_l} A_{l+1}$. Additionally, in the underlying graph G, for any edge $g = (u, v)$ with $\psi(g) = R_{\frac{l+1}{2}}$, we add a new node, E_g and 2 new edges: $e_1 = (u, E_g)$ and $e_2 = (E_g, v)$. We additionally assign $\varphi(E_g) = E$, $\psi(e_1) = R_E$, and $\psi(e_2) = R_F$. This procedure allows us to transform any odd path into an even path, giving a definition for the non-normalized HeteSim score h for odd length paths.

As a final step, HeteSim is normalized so that the normalized score for any two nodes lies in the interval [0, 1]. To do so, we will cast the problem in the language of transition matrices.

Definition 6. *Given a relation $A \xrightarrow{R} B$, let W_{AB} be an adjacency matrix between type A and type B. Let U_{AB} be W_{AB} normalized along each row vector. That is, U_{AB} is the transition probability matrix $A \to B$ based on relation R where each allowed transition is given equal probability. Similarly, let V_{AB} be a normalized form of the matrix W_{AB}, this time normalized along its column vectors. So, V_{AB} is the transition probability matrix for $B \to A$ based on relation R^{-1}. Note that $\mathcal{U}_{AB} = V_{BA}^T$.*

Definition 7. *Given a metapath $\mathscr{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, the reachable probability matrix PM for that metapath is given by*

$$PM_{\mathscr{P}} = \mathcal{U}_{A_1A_2}\mathcal{U}_{A_2A_3}\cdots\mathcal{U}_{A_lA_{l+1}}.$$

Note that $PM_{\mathscr{P}}(i, j)$ gives us the probability of object $i \in A_1$ reaching object $j \in A_{l+1}$ under the path $\mathscr{P}$, under the assumption that at each step all valid transitions have equal probability.

The following lemma is implicit in [12], but it is stated here for clarity.

Lemma 1. *Let* $s \in A_1$, $t \in A_{l+1}$. *Let* $\mathscr{P} = (A_1 A_2 \ldots A_{l+1})$ *be a metapath. Then,*

$$h(s, t \mid \mathscr{P}) = PM_{\mathscr{P}_L}(s, :)\left(PM_{\mathscr{P}_R}^{-1}(t, :)\right)^T,$$

where $PM_{\mathscr{P}_L}(a, :)$ *is used to denote the ath row of the matrix* $PM_{\mathscr{P}}$, *and* $\mathscr{P} = \mathscr{P}_L \mathscr{P}_R$ *is the decomposition of* $\mathscr{P}$ *into two paths of equal length.*

Proof. First, we only need to prove this result for even values of l. We proceed by induction. In the base case, we have $l = 0$. This is the trivial metapath, and its corresponding relation is the self relation. We have

$$h(s, s) = \delta(s, s) = 1,$$

and

$$PM_{\mathscr{P}_L}(s, :)\left(PM_{\mathscr{P}_R}^{-1}(s, :)\right)^T = 1 \times 1 = 1.$$

Therefore, the base case holds.

For the induction step, let $k \geq 2$ be an even integer. Assume that the lemma holds for all metapaths of length k. We will prove the lemma for paths of length $k + 2$. Beginning with the definition of h, we have

$$\begin{aligned} &h(s, t \mid R_1 \circ R_2 \circ \cdots \circ R_{k+2}) \\ &= \frac{1}{|O(s \mid R_1)||I(t \mid R_{k+2})|} \sum_{a \in O(s \mid R_1)} \sum_{b \in I(t \mid R_{k+2})} h(a, b \mid R_2 \circ \cdots \circ R_{k+1}) \\ &= \frac{1}{|O(s \mid R_1)||I(t \mid R_{k+2})|} \sum_{a \in O(s \mid R_1)} \sum_{b \in I(t \mid R_{k+2})} PM_{\mathscr{P}'_L}(a, :)\left(PM_{(\mathscr{P}')_R}^{-1}(b, :)\right)^T, \end{aligned}$$

where $\mathscr{P}' = R_2 \circ \cdots \circ R_{k+1}$, and the second equality follows from the induction hypothesis. Recalling the interpretation of $PM_{\mathscr{P}}$ as the product of transition matrices, we see

$$\begin{aligned} &\frac{1}{|O(s \mid R_1)||I(t \mid R_{k+2})|} \sum_{a \in O(s \mid R_1)} \sum_{b \in I(t \mid R_{k+2})} PM_{\mathscr{P}'_L}(a, :)\left(PM_{(\mathscr{P}')_R}^{-1}(b, :)\right)^T \\ &= \sum_{a \in O(s \mid R_1)} \frac{1}{|O(s \mid R_1)|} PM_{\mathscr{P}'_L}(a, :) \sum_{b \in I(t \mid R_{k+2})} \frac{1}{|I(t \mid R_{k+2})|}\left(PM_{(\mathscr{P}')_R}^{-1}(b, :)\right)^T \\ &= \left(\mathscr{U}_{A_1 A_2} PM_{\mathscr{P}'_L}(s, :)\right)\left(V_{A_{k+1} A_{k+2}} PM_{(\mathscr{P}')_R}^{-1}(t, :)\right)^T \\ &= PM_{\mathscr{P}_L}(s, :)\left(PM_{\mathscr{P}_R}^{-1}(t, :)\right)^T, \end{aligned}$$

which establishes the result. □

Finally, the HeteSim score is given by the cosine of the angle θ defined by vectors $PM_{\mathscr{P}_L}(s,:)$ and $PM_{\mathscr{P}_R^{-1}}(t,:)$.

Definition 8. *The normalized HeteSim score between two objects a and b based on the relevance path $\mathscr{P}$ is*

$$HS(s,t \mid \mathscr{P}) = cos(\theta) = \frac{PM_{\mathscr{P}_L}(s,:)\left(PM_{\mathscr{P}_R^{-1}}(t,:)\right)^T}{\left|PM_{\mathscr{P}_L}(s,:)\right|\left|\left(PM_{\mathscr{P}_R^{-1}}(t,:)\right)^T\right|}.$$

The above definition uses the multiplication of transition matrices to obtain reachable probability matrices, which in turn give the HeteSim score with respect to a given metapath. We can recast this matrix multiplication in the language of random walks. Consider the example graph and metapath given in Figure 1. Beginning with node s, we assign the probability value 1, since this is the specified source node. Next, we distribute that probability among all neighbors of s with type A_2 joined by an edge of type R_1. These neighbors are a, b and c, and each of these three nodes gets labeled with the probability 1/3. We repeat the same process with the neighbors of a, b, c having type A_3 and joined by an edge of type R_2. The probability 1/3 assigned to node a is split between its neighbors d and f, with each neighbor receiving 1/6. Node b has no eligible neighbors, and so its probability mass does not propagate to the next layer of the graph. Node c splits its probability mass of 1/3 between d and e. Therefore, d is labeled with probability mass 1/3, with 1/6 coming from a and 1/6 from c. Node e only receives probability mass from c and is therefore labeled with 1/6. Similarly, node f receives probability mass only from a, and therefore has total probability mass 1/6. This computation, which is equivalent to the matrix multiplication described above, gives

$$PM_{\mathscr{P}_L}(s,:) = \begin{bmatrix} 1/3 \\ 1/6 \\ 1/6 \end{bmatrix}.$$

To obtain $PM_{\mathscr{P}_R^{-1}}(t:)$, we repeat the same procedure on the second half of the metapath, this time working backwards towards A_3 from t. To start, t gets probability mass label 1. That probability is split among its 2 neighbors in A_4, giving g and h each probability mass 1/2. The mass of g is split evenly among d and e, so both of these nodes have probability mass 1/4. All of the probability mass of h goes to f, giving f a probability mass 1/2. Note that we have now labeled nodes d, e and f twice, once from the left and once from the right. While the labels from the left gave us $PM_{\mathscr{P}_L}(s:)$, the labels from the right give

$$PM_{\mathscr{P}_R^{-1}}(t,:) = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/2 \end{bmatrix}.$$

Finally, we can compute

$$HS(s,t \mid \mathscr{P}) = \frac{PM_{\mathscr{P}_L}(s,:)\left(PM_{\mathscr{P}_R^{-1}}(t,:)\right)^T}{\left|PM_{\mathscr{P}_L}(s,:)\right|\left|\left(PM_{\mathscr{P}_R^{-1}}(t,:)\right)^T\right|} = \frac{1/4}{1/2 \cdot \sqrt{6}/4} = \frac{\sqrt{6}}{3}.$$

1.6. Overview of SemNet's Existing HeteSim Implementation

The implementation of HeteSim in SemNet version 1 includes more than just the single-metapath HeteSim computation described in Section 1.5. In SemNet, HeteSim is not just used to give a score of the relatedness of two specific nodes with respect to a fixed metapath. Instead, it is used as a tool to rank a set of candidate source nodes based on their relatedness to a fixed target node.

Figure 2 gives an overview of this ranking algorithm as it exists in SemNet version 1. As input, the algorithm accepts a set of candidate source nodes S and a single target node t. In step 1, the set of all metapaths $\mathscr{MP}$ which have an instance joining some element of S to t is enumerated. This enumeration depends upon the underlying knowledge graph, which is stored in Neo4j. Step 2 is the computation of HeteSim scores for each triple (s, t, m) for $s \in S$, $m \in \mathscr{MP}$. For any fixed metapath $m \in \mathscr{MP}$, the results from step 2 induce a ranking on the source nodes S by HeteSim score. Step 3 takes these $|\mathscr{MP}|$ rankings and combines them to form a single ranking using a technique called ULARA (see [14]). Finally, this combined ranking is returned to the user and is used as an indication of which nodes from S are most closely related to t.

In this work, we will keep the overall structure of the HeteSim algorithm outlined in Figure 2, but will make several substantial changes to the various subroutines. First, we will replace the knowledge graph data structure using Neo4j with one based solely on Python dictionaries. Second, we will explore algorithms using randomization as candidate replacements for Step 2. Finally, we will discuss a flaw in ULARA and will replace Step 3 with the generation of a ranking based on mean HeteSim score over all metapaths. We will also explore an approximate version of Step 3 where only a subset of metapaths are selected for inclusion in the mean.

2. Methods

2.1. A New Method for Combining HeteSim Scores from Multiple Metapaths

SemNet version 1 outputs a ranking of many candidate source nodes with respect to a fixed target node. This ranking is intended to reflect the overall relatedness of each source node to the target node. SemNet version 1 computes the HeteSim scores for all requested source nodes and for all possible metapaths (up to some length bound) joining those source nodes to the target node. Each metapath induces a ranking of the source nodes according to HeteSim score. In order to combine these many rankings into a single ranking, SemNet version 1 uses a technique called ULARA (Unsupervised Learning Algorithm for Rank Aggregation) [14]. Due to a flaw in ULARA, this work replaces ULARA with a ranking based on mean HeteSim scores.

2.1.1. Background on ULARA—ULARA (Unsupervised Learning Algorithm for Rank Aggregation) [14] was developed by Klevmetiev et al. to solve the problem of *rank aggregation*. Rank aggregation considers the question of how to combine multiple rankings of a set of objects. Consider, for example, the problem of combining the results of multiple search engines into a single "best" ranking. Each search engine gives a different ordering of results. When the search engines disagree on which items are more relevant than other items, it is not immediately clear how to resolve this discrepancy and output a "best" ordered list of search results. ULARA proposes one solution to this problem based on an optimization problem. Conceptually, ULARA computes with mean rank of each object. The algorithm then finds a linear combination of the input ranking functions, giving more weight to functions that agree more closely with the mean ranking.

We now move to a formal mathematical exposition of ULARA. Note that we explain ULARA in the full generality with which it is presented in [14], but SemNet version 1 does not require the full generality of ULARA and may be thought of as using a special case.

Let X be a set of objects to be ranked, and let Q be a set of valid queries. Let $x, x' \in X$, $q \in Q$. Let $r: Q \times X \to \mathbb{N}$ be a ranking function, so that $r(q, x) < r(q, x')$ means that x has a higher ranking than x' with respect to the query q. Let $N \in \mathbb{N}$. Given a set of ranking functions $\{r_i\}_{i=1}^{N}$, ULARA produces a ranking function of the form

$$R(q, x) = \sum_{i=1}^{N} w_i r_i(q, x),$$

for some real numbers $\{w_i\}_{i=1}^{N}$ satisfying $0 \leq w_i \leq 1$ for all $1 \leq i \leq N$ and $\sum_{i=1}^{N} w_i = 1$. The value of each w_i is determined by an optimization problem. Let

$$\mu(q, x) = \frac{\sum_{i: r_i(q, x) \leq \kappa_i} r_i(q, i)}{|\{i: r_i(q, x) \leq \kappa_i\}|'}$$

where κ_i is a threshold which allows for the possibility that not every ranking function returns a rank for every $x \in X$. The function $\mu(q, x)$ is intended to represent the mean ranking of element x with respect to query q over all ranking functions r_i. Let

$$\sigma_i = (r_i(q, x) - \mu(q, x))^2.$$

This variance-like function is used to measure the agreement of ranking functions with each other, with the goal of giving ranking functions that agree with the mean a higher weight. Let

$$\delta_i(q, x) = w_i \sigma_i(q, x).$$

We can now finally state the optimization problem at the center of ULARA:

$$\underset{w_1,\ldots,w_N}{\arg\min} \sum_{q\in Q}\sum_{x\in X}\sum_{i=1}^{N}\delta_i(q,x),$$

subject to the constraints

$$\sum_{i=1}^{n} w_i = 1 \text{ and } \forall i, w_i > 0.$$

Note that this optimization problem is intended assign more weight to the ranking functions that agree most closely with the average ranking. ULARA solves the optimization problem using gradient descent. The details of the gradient descent algorithm are not relevant to the conceptual flaw in ULARA and are not presented here.

2.1.2. A Flaw in ULARA—The flaw in ULARA can be seen simply by examining the optimization problem itself. Let

$$a_i = \sum_{q\in Q}\sum_{x\in X}\sigma_i(q,x).$$

Then, the optimization problem becomes

$$\underset{w_1,\ldots,w_N}{\arg\min} \sum_{i=1}^{N} w_i a_i,$$

subject to the constraints

$$\sum_{i=1}^{N} w_i = 1 \text{ and } \forall i, w_i > 0.$$

Let j be such that $a_j = \min_i a_i$. Then, an optimal solution is given by

$$w_i = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}.$$

Further, the solution is unique if a_j is the unique minimum of the set $A = \{a_1, \ldots, a_N\}$. The case where the optimization problem does not have a unique solution is not mentioned in [14], and it seems this case should be rare in practice. Therefore, any unique optimal solution of the ULARA optimization problem places all of the available weight on a single ranking function. That is, ULARA does not give an aggregation of ranking functions; it simply selects a single raking function which shows most agreement with the others. In the language of SemNet, this should mean that only one metapath is used to give the final ranking of source nodes.

2.1.3. Implications for SemNet—Despite the fact that the math shows that only one metapath should have been used to generate rankings in SemNet version 1, this is not what actually happened. If only 1 metapath had actually been used to compute the rankings for SemNet version 1, it would be seemingly impossible that the produced ranking results would make sense. Yet, in multiple cases examined by domain experts in various fields (Alzheimer's disease, amyotrophic lateral sclerosis, leukemia, SARS coronavirus, and many more), the SemNet version 1 ranking results were quite intuitive. Thus, it was necessary to reconcile how the produced SemNet version 1 rankings would appear generally accurate despite the identified flaw in the original ULARA algorithm published by Klementiev and colleagues [14]. As such, a line by line examination of the adopted implementation of ULARA in the actual SemNet version 1 code [13] was performed and compared to the original published ULARA implementation [14]. The careful evaluation of the adopted ULARA implementation in SemNet version 1 identified a previously unseen but helpful coding bug that partially fixed the issue with the original ULARA. Specifically, the code in SemNet version 1 resulted in the ULARA algorithm terminating before the gradient descent had converged. As a result, a linear combination of multiple ranking functions (with nonzero coefficients) was actually returned, and multiple metapaths therefore are reflected in the rankings given by SemNet. Thus, unlike the original and above described ULARA, which would have only used 1 metapath to perform the ranking, the helpful bug in the ULARA implementation within SemNet version 1 used a partially averaged ranking that contained multiple metapaths. As such, SemNet version 1 was still able to be used by domain scientists to produce helpful and seemingly sensible rankings. While the serendipitous bug rendered SemNet version 1 useful, a fundamentally correct replacement for the ULARA algorithm is necessary.

As a replacement for ULARA, in SemNet version 2, the mean HeteSim score of a source node with respect to all metapaths is used to generate a ranking of source nodes.

2.2. Computational Analysis of HeteSim Runtimes: SemNet Version 1

To better understand the runtime of the HeteSim computation, the Python module time [36] was used to record the time required to compute HeteSim for each of the metapaths from the studied source nodes to Alzheimer's disease. Additionally, the total time spent on the required Neo4j queries was recorded for each metapath. This allows separate analysis of the time required to query the graph and the time required to perform the HeteSim computations.

2.3. Development, Implementation, and Testing of Algorithms

The core development work for this project can be divided into three general categories: re-implementation of the knowledge graph data structure, development and implementation of algorithms, and testing.

2.3.1. Knowledge Graph Data Structure—SemNet version 1 used Neo4j to store the knowledge graph. After preliminary testing showed that Neo4j was likely a significant bottleneck, the knowledge graph data structure was re-implemented using nested Python dictionaries. Because these dictionaries use hashing for lookup, they have average lookup

time $O(1)$ (see, e.g., [37]). As a result, dictionaries allow for quickly examining the neighborhood of a node in the knowledge graph, restricted to edge and node types of interest. Consequently, it is also efficient to traverse paths within the graph.

After testing on artificial examples, a knowledge graph object was built using an edge set derived from SemMedDB. This is an updated version of the edge set, and is not identical to the edge set from SemNet version 1.

2.3.2. Development of Approximation Algorithms—In addition to the data structure improvements, approximation algorithms based on randomization were explored as a way of further increasing performance. In particular, approximation algorithms were investigated as possible replacements for the computation of HeteSim on a single metapath (step 2 in Figure 2) and aggregation of rankings (step 3 in Figure 2).

2.3.3. Implementation and Testing—All code were implemented in Python 3. Testing was performed using Jupyter Notebook 5.5.0 [38] and Python 3.6.10 [39]. All code were run on a server with 1 NVIDIA TESLA v100 GPU with 32 GB RAM and a 48 core CPU with 320 GB RAM.

For all code not involving randomization, the correctness of implementation was assessed using unit tests, which may be found in the source code repository. The one randomized function of significant complexity, randomized pruned HeteSim, was assessed on artificially-constructed example knowledge graphs. These examples were constructed by hand by the authors, and the full examples may be found in the source code repository. The algorithm was run on each graph 100 times with parameters $\epsilon = 0.05$ and $r = 0.95$. As with the SemNet version 1 implementation, the speed of the new implementation was assessed using the Python time module [36].

2.3.4. User Study Methods—A small user study was performed to quantify the significant differences between two groups of users: a group of naive SemNet version 1 users (n = 11) and a group of naive SemNet version 2 users (n = 10) to determine how many users were comfortable in running a simulation after a short standardized training session that also included reading the user documentation. To ensure degree of previous Python experience was not biasing the analysis, groups were selected to ensure equivalent distributions of prior Python user experience. Additionally, a third group of users (n = 7) trained in both SemNet version 1 and SemNet version 2 was used to compare the user friendliness of SemNet version 1 and version 2. A simple categorical standardized electronic survey was used to quantify comfort in using SemNet version 2 and its user friendliness. Details are provided in the Results in Section 3.4. Fisher's exact test was used to perform statistical analysis in Microsoft Excel.

3. Results

3.1. Computational Analysis of HeteSim Runtimes: SemNet Version 1

For each of the three source nodes, the runtime of the HeteSim computation on each metapath from the source node to Alzheimer's disease was recorded. The computation time

results are given in Table 1, and the distribution of runtimes is depicted graphically in Figure 3. Note that SemNet version 1 incorporated parallelization, allowing multiple HeteSim computations for different metapaths to occur simultaneously. Therefore, the computation time per metapath times the number of metapaths does not equal the total computation time. Time required for the neo4j graph queries was also measured and is displayed in Figure 4.

3.2. Algorithms

In this section, we present several algorithms for computing HeteSim and variants. Proofs of correctness are also given where appropriate.

We consider two main algorithms for computing HeteSim on a single metapath and two algorithms for aggregating HeteSim scores across multiple metapaths. For computing HeteSim on a single metapath, we consider the deterministic HeteSim algorithm used in SemNet version 1 and a new algorithm, randomized pruned HeteSim. For aggregating HeteSim scores over multiple metapaths, we consider computing the exact mean over all metapaths and also an algorithm which approximates the mean by taking the mean over a random subset of metapaths. We also combine these algorithms to obtain three distinct algorithms for computing (an approximation to) the mean HeteSim score: deterministic HeteSim with exact mean, deterministic HeteSim with approximate mean, and randomized pruned HeteSim with approximate mean. Using approximate mean HeteSim as an example, an overview of the new algorithm structure, emphasizing changes, is shown in Figure 5.

3.2.1. Deterministic HeteSim—For completeness, we summarize the deterministic algorithm for computing HeteSim. While this same algorithm is used in SemNet version 1, SemNet version 2 significantly improves the implementation by changing the underlying data structure for the knowledge graph. Where version 1 used Neo4j, version 2 uses a knowledge graph object built from Python dictionaries.

Given a source node s, a target node t, and a metapath $\mathscr{P}$, the deterministic HeteSim algorithm begins by splitting $\mathscr{P}$ into two halves: $\mathscr{P}_L$ and $\mathscr{P}_R$. If $\mathscr{P}$ has odd length, the construction described in Section 1.5 is applied before constructing $\mathscr{P}_L$ and $\mathscr{P}_R$. An identical subroutine is now applied to both $\mathscr{P}_L$ and $\mathscr{P}_R^{-1}$. The following exposition will consider only $\mathscr{P}_L$.

Recall that the algorithm must compute $PM_{\mathscr{P}_L}(s, :)$, which may be understood as the probability that a random walk along the given metapath starting from s arrives at a given node in $A_{l/2}$. The algorithm iteratively computes the probability of arriving at each node in A_i for step i of the metapath for $1 \leq i \leq l/2$.

Let $v_i(x)$ be the probability of arriving at node x of type A_i at step i of the metapath. To compute v_i for $i > 1$, note that it is sufficient to know v_{i-1}, as

$$v_i(x) = \sum_{y \in \delta^-_{R_i-1}(x)} \frac{1}{\delta^+_{R_i-1}(y)} v_{i-1}(y).$$

Therefore, beginning with $v_1(s) = 1$, the algorithm iteratively computes v_2, …, $v_{l/2}$ and $PM_{\mathscr{P}_L} = v_{l/2}$. After completing the analogous computation for $\mathscr{P}_R^{-1}$, the algorithm returns

$$\frac{PM_{\mathscr{P}_L}(a,:)\left(PM_{\mathscr{P}_R^{-1}}(b,:)\right)^T}{\left|PM_{\mathscr{P}_L}(a,:)\right|\left|\left(PM_{\mathscr{P}_R^{-1}}(b,:)\right)^T\right|}.$$

Pseudocode is given in Algorithms 1 and 2.

Algorithm 1: HeteSim.

Input: start node s, end node t, metapath $\mathcal{P}$ of even length {odd relevance paths must be preprocessed}
Output: HeteSim score
Construct $\mathcal{P}_L, \mathcal{P}_A$
$v_L \leftarrow oneSidedHS(s, \mathcal{P}_L)$
$v_R \leftarrow oneSidedHS(t, \mathcal{P}_R^{-1}$
return $(v_L \cdot v_R)/(\|v_L\|v_R\|)$

Algorithm 2: oneSidedHS subroutine.

Input: start node s, metapath $\mathcal{P}$
Output: vector $v_{\text{length}(\mathcal{P})}$, the one-sided HeteSim vector
for $i = 1$ to length$(\mathcal{P})/2$ **do**
$v_i \leftarrow [0]^{\|A_i\|}$ {Vector of zeros, indexed by elements of A_i}
end for
$v_1[s] = 1$
for $i = 2$ to length$(\mathcal{P})$ **do**
for $x \in A_i$ **do**
$v_i[x] \leftarrow \sum_{y \in \delta_{R_{i-1}^-}} \frac{1}{\delta^+_{R_{i-1}}[y]} v_{i-1}[y]$
end for
end for
return $v_{\text{length}(\mathcal{P})}$

3.2.2. Pruning the Graph—

Given a metapath $\mathscr{P}_{\mathscr{L}} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_{\frac{l}{2}-1}} A_{\frac{l}{2}}$, a random walk starting from 2 $s \in A_1$ may arrive at node $u \in A_i$ such that the out degree of u along edges of type R_i is 0. Informally speaking, the random walk has reached a dead end. As an example, node b in Figure 1 is a dead end. The presence of these dead ends reduces the probability that a random walk starting from s actually reaches any node of type $A_{\frac{l}{2}}$. In fact, we can construct graphs that make this probability arbitrarily small. Therefore, a basic random walk algorithm may have arbitrarily long runtime. We will address this limitation by defining a new but closely related quantity: pruned HeteSim.

Before proceeding, we provide two additional examples to explore the effect of dead ends on HeteSim scores. In Figure 6, a simple knowledge graph is shown, organized according to one metapath. The nodes are organized into columns by type, and the columns are given in the order that those types appear in the metapath. The only edges shown are those which appear in some instance of the metapath. This graph has $m_1 - 1$ dead-end nodes on the left-hand side and $m_2 - 1$ dead-end nodes on the right-hand side. We can compute its HeteSim score as follows.

$$\text{HS}(s, t \mid \mathscr{P}) = \frac{1 \cdot 1}{1 \cdot 1} = 1.$$

Note that this score does not change with m_1 or m_2. In particular, the HeteSim score with the given graph is identical to the HeteSim score when all dead ends are removed from the graph. As we will later see, this result generalizes to all metapaths of length less than or equal to 4.

In contrast, the metapath and knowledge graph depicted in Figure 7 create a situation where the removal of dead ends does change the HeteSim score. If we take $m = 2$, then we have removed all dead-end nodes. In this case, the HeteSim score is

$$\mathrm{HS}(s,t \mid \mathscr{P}) = \frac{[3/4\ \ 1/4]([1/2\ \ 1/2])^T}{\|[3/4\ \ 1/4]\|\|[1/2\ \ 1/2]\|} = \frac{1/2}{\sqrt{5/8}\sqrt{1/2}} = \frac{2\sqrt{5}}{5}.$$

If we instead take $m = 3$, then the HeteSim score is $\frac{5\sqrt{34}}{34}$, and, in the limit as $m \to \infty$, the HeteSim score approaches $\frac{\sqrt{2}}{2}$.

We now introduce a new score: Pruned HeteSim. This new score is identical to HeteSim on relevance paths of length at most 4. To rigorously define Pruned HeteSim, we must first formally define a dead-end node at step i of a given metapath and with respect to nodes s and t.

Let $G = (V, E)$ be a heterogeneous information network, and let $\mathscr{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{l+1}$ be a metapath in G. Let $s \in V$ with $\psi(s) = A_1$ and $t \in V$ with $\psi(t) = A_l$. Let C_1 be the set of nodes of type $A_{l/2}$ reachable from s along metapath $\mathscr{P}_L$. Similarly, let C_2 be the set of nodes of type $A_{l/2}$ reachable from t along metapath $\mathscr{P}_R^{-1}$. Let $C = C_1 \cap C_2$, and label the elements of C so that $C = \{c_1, c_2, \ldots, c_j\}$. For $i \leq j$, let X_i be the event that a random walk starting at s along $\mathscr{P}_L$ ends at node c_i. Similarly, let Y_i be the event that a random walk starting at t along $\mathscr{P}_R^{-1}$ ends at node c_i. Let $x_i = Pr(X_i)$ and $y_i = Pr(Y_i)$. Let $x = (x_1, x_2, \ldots, x_j)$ and let $y = (y_1, y_2, \ldots, y_j)$.

Let Z be the event that a random walk starting from s along $\mathscr{P}_L$ reaches some node in C. Similarly, let W be the event that a random walk starting from t along $\mathscr{P}_R^{-1}$ reaches some node in C.

Definition 9. *For a node v belonging to any of $A_1, A_2, \ldots, A_{l/2}$, we define a dead end as follows. Let metapath $\mathscr{P}$ and source node s be fixed. Let A be the event that a random walk beginning from s and following metapath $\mathscr{P}_L$ contains node v at step i (so that the type of v is A_i). Then, v is a* dead end *at step i of metapath $\mathscr{P}$ and with respect to source node s if and only if $Pr(Z|A) = 0$. For a node w belonging to any of $A_{l/2+1}, \ldots, A_{l+1}$, the definition is analogous. Let metapath $\mathscr{P}$ and target node t be fixed. Let B be the event that a random walk starting from t and following metapath $\mathscr{P}_R^{-1}$ contains node w at step i. Then, w is a* dead end *with respect to step i of metapath, $\mathscr{P}$ and target node t if and only if $Pr(W|B) = 0$. For fixed*

nodes s, t and fixed metapath $\mathscr{P}$, let D_i be the set of dead-end nodes at step i of metapath $\mathscr{P}$ with respect to source node s and target node t.

Informally, this definition means that a node v is a dead end at step i of a metapath if no random walk which reaches the set of central nodes C has v as its ith node. Recall that non-normalized HeteSim is defined by

$$h(s,t \mid R_1 \circ R_2 \circ \cdots \circ R_l) = \frac{1}{|O(s \mid R_1)||I(t \mid R_l)|} \sum_{a \in O(s \mid R_1)} \sum_{b \in I(t \mid R_l)} h(a,b \mid R_2 \circ R_3 \circ \cdots \circ R_l - 1),$$

where $O(s|R_1)$ is the set of out-neighbors of node s based on relation R_1, and $I(t|R_l)$ is the set of in-neighbors of node t based on the relation R_l. To define the non-normalized version of pruned, we simply exclude dead-end nodes from the sets of neighbors.

Definition 10. *Let $\mathscr{P} = R_1 \circ R_2 \circ \cdots \circ R_l$ be a metapath in some graph G. Let s, t belong to the vertex set of G, and let D_i be the set of dead-end nodes at step i of metapath $\mathscr{P}$. Then, the non-normalized pruned HeteSim score is given by*

$$g(s,t \mid R_1 \circ R_2 \circ \cdots \circ R_l) = \frac{1}{|O(s \mid R_1) \setminus D_1||I(t \mid R_l) \setminus D_l|} \sum_{a \in O(s \mid R_1) \setminus D_1} \sum_{b \in I(t \mid R_l) \setminus D_l} h(a,b \mid R_2 \circ R_3 \circ \cdots \circ R_l - 1),$$

where $O(s|R_1)$ is the set of out-neighbors of node s based on relation R_1, and $I(t|R_l)$ is the set of in-neighbors of node t based on the relation R_l.

The normalization of pruned HeteSim proceeds exactly like that for HeteSim. We obtain a restricted adjacency matrix $W'_{AB,i}$ for the relation $A \xrightarrow{R_i} B$ by removing any 1s in W_{AB} corresponding to a dead-end node in B at step i of the metapath. As before, we normalize $W'_{AB,i}$ along its row vectors to obtain $\mathscr{U}'_{AB,i}$. As before, we can obtain a reachable probability matrix by multiplying the normalized restricted adjacency matrices:

$$PM'_{\mathscr{P}} = \mathscr{U}'_{A_1A_2,2}\mathscr{U}'_{A_2A_3,3}\cdots\mathscr{U}'_{A_lA_{l+1},l+1}.$$

Definition 11. *The normalized pruned HeteSim score is given by*

$$PHS(a,b \mid \mathscr{P}) = \frac{PM'_{\mathscr{P}_L}(a,:)\left(PM'_{\mathscr{P}_R^{-1}}(b,:)\right)^T}{\sqrt{\left|PM'_{\mathscr{P}_L}(a,:)\right|\left|\left(PM'_{\mathscr{P}_R^{-1}}(b,:)\right)^T\right|}}.$$

Note that, for metapaths with no repeated node types, pruned HeteSim may be computed by simply removing all dead-end nodes from the graph and then computing HeteSim on this pruned graph. Importantly, pruned HeteSim has value equal to plain HeteSim for metapaths of length at most 4. Since these shorter paths are often the ones of most interest in

small-diameter knowledge graphs, pruned HeteSim may be thought of as a replacement for HeteSim in these circumstances.

Additionally, note that Definition 11 gives rise to a deterministic algorithm for computing pruned HeteSim, much like the deterministic algorithm for HeteSim. The algorithm now requires 2 passes over the data structure. In the first pass over the data, dead ends are identified. In a second pass, Definition 11 allows for the computation of the non-normalized pruned HeteSim score. Normalization is applied as the final step. Because our computational focus in this manuscript is on short paths of length at most four, and because HeteSim and pruned HeteSim have the same values for paths of length at most four, we do not pursue the deterministic algorithm for pruned HeteSim further. For these short paths, a deterministic computation of HeteSim is faster than a deterministic computation of pruned HeteSim.

Theorem 1. *Let* $\mathscr{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{l+1}$ *be a metapath with length* $l \leq 4$. *Then*,

$$PHS(s, t \mid G, \mathscr{P}) = HS(s, t \mid G, \mathscr{P}).$$

Proof. First, note that we only need to consider metapaths with even length, as odd metapaths will simply be transformed to even length metapaths before HeteSim is computed. Next, note that the result is trivial for metapaths with length 2, as these can have no dead ends. We may therefore focus only on the case where the metapath has length 4.

Let $\mathscr{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} A_3 \xrightarrow{R_3} A_4 \xrightarrow{R_4} A_5$ be a metapath in G. Note that there can be no dead ends of type A_3. Additionally, if s or t is a dead end, then $\text{HS}(s, t \mid G, \mathscr{P}) = 0 = \text{PHS}(s, t \mid G, \mathscr{P})$. Therefore, we may assume that all dead ends are of type A_2 or A_4.

Recall that X_i is the event that a random walk in G from s reaches node c_i, and similarly Y_i is the event that a random walk in G starting at t arrives at node c_i. Let X_i' be the event that a random walk in G' along metapath $\mathscr{P}_L$ starting from s arrives at node c_i. Similarly let Y_i' be the event that a random walk in G' along metapath $\mathscr{P}_R^{-1}$ arrives at node c_i. Let p_L be the probability that a random walk starting from s arrives at a dead-end node in A_2. Similarly, let p_R be the probability that a random walk beginning at t will arrive at a dead end in A_4. Note that, once a random walk has reached a non-dead-end node of type A_2 or A_4, that random walk must reach some node of type A_3. Therefore,

$$Pr(X_i) = (1 - p_L) Pr(X_i')$$

and

$$Pr(Y_i) = (1 - p_R) Pr(Y_i').$$

Letting $x_i = Pr(X_i)$, $y_i = Pr(Y_i)$, $x_i' = Pr(X_i')$, and $y_i' = Pr(Y_i')$, observe

$$\begin{aligned}\mathrm{HS}(s,t \mid G,\mathscr{P}) &= \frac{\sum_{i=1}^{k} x_i y_i}{\sqrt{\sum_{i=1}^{k} x_i^2 \sum_{i=1}^{k} y_i^2}} \\ &= \frac{\sum_{i=1}^{k} (1-p_L) x_i' (1-p_R) y_i'}{\sqrt{\sum_{i=1}^{k} (1-p_L)^2 (x_i')^2 \sum_{i=1}^{k} (1-p_R)^2 (y_i')^2}} \\ &= \frac{\sum_{i=1}^{k} x_i' y_i'}{\sqrt{\sum_{i=1}^{k} (x_i')^2 \sum_{i=1}^{k} (y_i')^2}} \\ &= \mathrm{PHS}(s,t \mid G,\mathscr{P}).\end{aligned}$$

□

3.2.3. Pruned HeteSim—We now present an alternate algorithm for computing a variant of the HeteSim score. This algorithm is much more computationally tractable, and we have shown that the HeteSim and pruned HeteSim scores are identical for relevance paths of length at most 4.

Let $\mathscr{P}$ be a metapath, and let s and t be source and target notes, respectively. Let N be a positive integer, the required value of which will be determined later. Starting from s the algorithm takes N random walks along $\mathscr{P}_R$, never visiting any node that has been marked as a dead end for the current step of the metapath. At any point, if the algorithm encounters a dead end, it marks the current node as a dead end for the current step of the metapath and then retraces its steps until a non-dead-end node is reached, marking dead ends along the way as necessary. Note that any dead end at a given step in the metapath will only need to be marked once, and the algorithm will avoid it for all future random walks. The same algorithm is repeated along metapath $\mathscr{P}_L^{-1}$ starting from t.

The frequency vectors of the terminal nodes of the random walks give an approximation for $PM'_{\mathscr{P}_L}$ and $PM'_{\mathscr{P}_R^{-1}}$, which are used to approximate the pruned HeteSim score. Psuedocode is given in Algorithms 3–5. Analysis of the algorithm, determination of N, and a formal proof of correctness are given in Section 3.2.4.

Algorithm 3: Randomized Pruned HeteSim.

Input: start node s, end node t, relevance path $\mathscr{P}$ of even length, error tolerance ϵ, success probability r {odd relevance paths must be preprocessed}
Output: approximate HeteSim score
$S \leftarrow$ breadthFirstSearch(s, $\mathscr{P}_L$)
$T \leftarrow$ breadthFirstSearch(t, $\mathscr{P}_R^{-1}$)
$k \leftarrow |S \cup T|$
$c \leftarrow (5+4\sqrt{2})/4$
$C \leftarrow 2(c+\sqrt{c^2+2\epsilon})^2 + \epsilon(c+\sqrt{c^2+2\epsilon})$
$N \leftarrow \lceil \frac{C}{\epsilon^2} \rceil k \ln(4k/(1-r))$
return RandomizedPrunedHeteSimGivenN(s, t, $\mathscr{P}$, N)

Algorithm 4: RandomizedPrunedHeteSimGivenN subroutine.

Input: start node s, end node t, relevance path $\mathcal{P}$ of even length, number of iterations N
Output: approximate HeteSim score
for $i = 0$ to length($\mathcal{P}_L$)) **do**
  B[i] $\leftarrow \emptyset$
end for
$v_L \leftarrow [0]^k$ {array of 0s indexed by elements of K}
$v_R \leftarrow [0]^k$
{random walks from s}
for $n = 1$ to N **do**
  $(B, x) \leftarrow$ restrictedRandomWalkOnMetapath(s, $\mathcal{P}_L$, B)
  $v_L[x] = v_L[x] + 1$
end for
{random walks from t}
for $i = 0$ to length($\mathcal{P}_L$)) **do**
  B $\leftarrow \emptyset$
end for
for $n = 1$ to N **do**
  $(C, x) \leftarrow$ restrictedRandomWalkOnMetapath(t, $\mathcal{P}_R^{-1}$, B)
  $v_R[x] \leftarrow v_R[x] + 1$
end for
{compute approximate probability vectors and approximate pruned HeteSim}
$v'_L \leftarrow v_L / N$
$v'_R \leftarrow v_R / N$
return $(v'_L \cdot v'_R)/(||v'_L||v'_R||)$

Algorithm 5: restrictedRandomWalkOnMetapath subroutine.

Input: start node s, metapath $\mathcal{P}$, badNodes B
Output: (B, node), where node is the final node reached, and B is the updated list of dead-end nodes
$i \leftarrow 1$
nodeStack $\leftarrow$ [ ]
$x \leftarrow s$
while $i > 0$ **do**
  $Y \leftarrow$ neighbors(x, R_i) $\setminus B[i]$
  if $Y \neq \emptyset$ **then**
    {pick a neighbor with probability proportional to edge weight}
    $w \leftarrow \sum_{y \in Y}$ edgeweight(x, y)
    $z \leftarrow$ SelectWithProbability([(y, edgeWeight(x, y)/w) for $y \in Y$])
    nodeStack.push(x)
    $x \leftarrow z$
    $i \leftarrow i + 1$
  else
    {x is a dead end}
    B[i-1] $\leftarrow$ B[i-1] $\cup \{x\}$
    $x \leftarrow$ nodeStack.pop()
    $i \leftarrow i - 1$
  end if
end while
return (B, x)

3.2.4. Runtime Analysis of the Pruned HeteSim Algorithm—We now provide guarantee on the number of random walks required to approximate pruned HeteSim with a given error tolerance ϵ and success probability r.

Let $\mathcal{S}_k = \left\{v \in \mathbb{R}^k : \sum_i v_i = 1 \text{ and } v_i \geq 0\right\}$. We consider arbitrary $v, w \in \mathcal{S}_k$ for fixed k, where $v = PM'_{\mathcal{P}_L}(s, :)$ and $w = PM'_{\mathcal{P}_R^{-1}}(t, :)$. We will show that if all the entries in the vectors are sufficiently close to their true value, then the cosine will be sufficiently close to the true value. We consider $\hat{v}$, a random approximation of v after some number of steps. Notice that $\hat{v} = v + \lambda$, where $\lambda \in \mathbb{R}^k$ such that $\sum_{i=1}^k \lambda_i = 0$ and $v_i + \lambda_i \geq 0$ (since $\hat{v}$ is always a probability vector). Let

$$\mathcal{E}_k(v, \delta, \alpha, \beta) = \left\{ w \in \mathbb{R}^k : \sum_i w_i = 0, v_i + w_i \geq 0, v_i \geq \alpha \Rightarrow |w_i| \leq \delta |v_i|, \text{ and } v_i < \alpha \Rightarrow |w_i| \leq \beta\delta \right\}.$$

We now consider $\lambda \in \mathscr{E}_k(v, \delta, \alpha, \beta)$. Note that the bound imposed by $\mathscr{E}_k(v, \delta, \alpha, \beta)$ treats small entries and large entries in v differently. This will be important to achieve an $O(k \log k)$ bound on the number of required random walks N later in the section.

We start by giving sufficient conditions for a bound on $|\cos \theta' - \cos \theta|$, where θ' is the angle between $\hat{v}$ and $\hat{w}$ and θ is the angle between v and w.

Theorem 2. *Fix* $\epsilon > 0$. *Let* $0 \leq \beta, \bar{\beta} \leq 1$. *Let* $\alpha, \bar{\alpha} \geq 0$. *Let* $v, w \in \mathcal{S}_k$. *Let*

$$b = \frac{2 + \frac{k\beta^2}{2|v|^2}}{1 + \frac{1}{|v|\sqrt{k}}} + \sqrt{\frac{k\beta^2}{|v|^2} + 1}$$

and

$$a = \frac{\frac{k\beta^2}{|v|^2} + 1}{1 + \frac{1}{|v|\sqrt{k}}}$$

and

$$\bar{b} = \frac{2 + \frac{k\bar{\beta}^2}{2|w|^2}}{1 + \frac{1}{|w|\sqrt{k}}} + \sqrt{\frac{k\bar{\beta}^2}{|w|^2} + 1}$$

and

$$\bar{a} = \frac{\frac{k\bar{\beta}^2}{|w|^2} + 1}{1 + \frac{1}{|w|\sqrt{k}}}.$$

Let $\delta = \frac{\epsilon}{b + \sqrt{b^2 + 2a\epsilon}}$ *and* $\bar{\delta} = \frac{\epsilon}{\bar{b} + \sqrt{\bar{b}^2 + 2\bar{a}\epsilon}}$. *If* $\lambda \in \mathscr{E}_k(v, \delta, \alpha, \beta)$ *and* $\bar{\lambda} \in \mathscr{E}_k(w, \bar{\delta}, \bar{\alpha}, \bar{\beta})$ *then*

$$\left| \frac{(v + \lambda) \cdot (w + \bar{\lambda})}{|v + \lambda||w + \bar{\lambda}|} - \frac{v \cdot w}{|v||w|} \right| \leq \epsilon.$$

Proof. Follows from Lemma A5 in Appendix A and the triangle inequality. □

We now need to understand the probability that any given entry of $\hat{v}$ (or $\hat{w}$) is close to the corresponding entry of v (or w). Since the number of walks arriving at a given node is binomial, we apply a Chernoff bound (Lemma 2) to the binomial distribution to obtain Corollary 1.

Lemma 2 (Chernoff Bound [40]). *Let* $X \sim Binom(n, p)$. *Let* $\mu = \mathbb{E}(X) = np$. *For* $\delta > 0$,

$$Pr(X \leq (1-\delta)\mu) \leq exp\left(-\frac{\delta^2\mu}{2}\right)$$

and

$$Pr(X \geq (1+\delta)\mu) \leq exp\left(-\frac{\delta^2\mu}{2+\delta}\right).$$

Corollary 1. *Let* $X \sim Binom(n, p)$. *For* $\delta > 0$,

$$Pr\left(\left|\frac{X}{n} - p\right| > \delta p\right) \leq 2 \cdot exp\left(-\frac{n\delta^2 p}{2+\delta}\right)$$

and

$$Pr\left(\left|\frac{X}{n} - p\right| > \delta\right) \leq 2 \cdot exp\left(-\frac{n\delta^2}{2p+\delta}\right).$$

Having bounded the probability of any one vector entry having small error, we now use a union bound to bound the probability that all entries have small error.

Lemma 3. *Fix* $n, k \in \mathbb{N}$. *Fix* $\delta \geq 0$ *and* $0 \leq \alpha, \beta \leq 1$. *Let* $v = (v_1, \cdots, v_k)$ *such that* $v_i \geq 0$ *and* $\sum_i v_i = 1$. *Let* $X_i \sim Binom(n, v_i)$ *such that* $\sum_i X_i = n$. *Let* $\lambda_i = \frac{X_i}{n} - v_i$ *and let* $\lambda = (\lambda 1, \cdots, \lambda_k)$. *We have that*

$$Pr(\lambda \notin \mathscr{C}_k(v, \delta, \alpha, \beta)) \leq 2k\ exp\left(-n\delta^2 \cdot \min\left\{\frac{\beta^2}{2\alpha + \delta\beta}, \frac{\alpha}{2+\delta}\right\}\right)$$

Proof. Since $X_i \geq 0$, $v_i + \lambda_i \geq 0$. We now apply the Chernoff bound. For $v_i \geq \alpha$, we see that

$$Pr(|\lambda_i| \geq \delta v_i) = Pr\left(\left|\frac{X_i}{n} - v_i\right| \geq \delta v_i\right) \leq 2 \cdot \exp\left(-\frac{n\delta^2 v_i}{2+\delta}\right) \leq 2 \cdot \exp\left(-\frac{n\delta^2\alpha}{2+\delta}\right)$$

For $v_i < \alpha$, we see that

$$Pr(|\lambda_i| \geq \beta\delta) = Pr\left(\left|\frac{X_i}{n} - v_i\right| \geq \beta\delta\right) \leq 2 \cdot \exp\left(-\frac{n\beta^2\delta^2}{2v_i + \beta\delta}\right) \leq 2 \cdot \exp\left(-\frac{n\beta^2\delta^2}{2\alpha + \beta\delta}\right).$$

The result then follows by the union bound. □

Finally, we can combine the previous results to bound the required number of random walks, given error tolerance ϵ and success probability r.

Lemma 4. *Let* $\epsilon > 0$ *and* $0 < r < 1$. *For* $c(\epsilon) = 2\left(c + \sqrt{c^2 + 2\epsilon}\right)^2 + \epsilon\left(c + \sqrt{c^2 + 2\epsilon}\right)$ *and* $C = \frac{5 + 4\sqrt{2}}{4}$. *Let* δ *as in* Theorem 2. *After making n (non-dead-end) walks in the randomized pruned HeteSim algorithm*,

$$Pr\left(\lambda \notin \mathscr{E}_k\left(v, \delta, \frac{|v|}{\sqrt{k}}, \frac{|v|}{\sqrt{k}}\right)\right) \leq 2k\, exp\left(-\frac{n}{k} \cdot \frac{\epsilon^2}{c(\epsilon)}\right).$$

Proof. We apply Lemma 3 and Theorem 2. We set $\alpha = \beta = \frac{|v|}{\sqrt{k}} \leq 1$ and $\delta = \frac{\epsilon}{b + \sqrt{b^2 + 2a\epsilon}}$. Thus,

$$Pr(\lambda \notin \mathscr{E}_k(v, \delta, \alpha, \beta)) \leq 2k \exp\left(-n \cdot \frac{|v|}{\sqrt{k}} \cdot \frac{\epsilon^2}{2\left(b + \sqrt{b^2 + 2a\epsilon}\right)^2 + \epsilon\left(b + \sqrt{b^2 + 2a\epsilon}\right)}\right).$$

We notice that the content of the exponent is a decreasing function in $|v|$ (for $|v| > 0$). Thus,

$$Pr(\lambda \notin \mathscr{E}_k(v, \delta, \alpha, \beta)) \leq 2k \exp\left(-\frac{n}{k} \cdot \frac{\epsilon^2}{2\left(c + \sqrt{c^2 + 2\epsilon}\right)^2 + \epsilon\left(c + \sqrt{c^2 + 2\epsilon}\right)}\right).$$

□

Corollary 2. *Under the same assumptions as* Lemma 4, *let* $n > \frac{c(\epsilon)}{\epsilon^2} \cdot k \ln\left(\frac{4k}{1 - r}\right)$, *after making n (non-dead-end) walks in the randomized pruned HeteSim algorithm (on both sides of the computation)*,

$$Pr(\left|PHS(a, b \mid \mathscr{P}) - \widetilde{PHS}(a, b \mid \mathscr{P})\right| \epsilon) r\,.$$

Proof. Follows from Lemma 4 (applied to both sides of the computation), Theorem 2 and the union bound. □

Remark 1. *The exact number of walks required may differ due to the existence of walks that lead to dead end not counting. In* Appendix B, *we have provided some analysis of the probabilistic effects of this.*

3.2.5. Deterministic Aggregation—In order to rank the overall relatedness of source nodes to a fixed target node, SemNet version 2 uses the mean HeteSim score between the source and target node, averaged over all metapaths which exist for any source node in the set under study.

For completeness, pseudocode for computing exact mean HeteSim scores is given in Algorithm 6.

Algorithm 6: Exact Mean HeteSim score.

```
Input: set of start nodes S, end node t, path length p
Output: vector of mean HeteSim scores h, indexed by elements of S
  Construct M, the set of all metapaths between any element of S and t
  for s ∈ S do
    HSscores = []
    for m ∈ M do
      HSscores.append(HeteSim(s, t, m))
    end for
    h[s] = mean(HSscores)
  end for
  return h
```

3.2.6. Randomized Aggregation—As an alternative to taking the exact mean HeteSim score over all metapaths, we also consider an approximation to the mean given by the mean over a random subset of metapaths. Let S be a set of source nodes in the graph and T be a set of target nodes. Let $\mathcal{MP}_{ST}$ be the set of all metapaths in the knowledge graph with at least one instance between some node in S and some node in T.Let $(s, t) \in S \times T$. Let $\mathcal{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{+1}$ by a metapath. Recall that $\mathrm{HS}(s, t \mid \mathcal{P})$ is the HeteSim score between s and t relative to the metapath $\mathcal{P}$. Similarly, let $\mathrm{PHS}(s, t \mid \mathcal{P})$ be the Pruned HeteSim score between s and t relative to the metapath $\mathcal{P}$.

The aggregated HeteSim score of a source–target pair(s, t) is defined to be

$$Q(s,t) = \frac{1}{|\mathcal{MP}_{\mathcal{ST}}|} \sum_{\mathcal{P} \in \mathcal{MP}} \mathrm{HS}(s,t \mid \mathcal{P})$$

and the aggregated Pruned HeteSim Score is defined to be

$$R(s,t) = \frac{1}{|\mathcal{MP}|} \sum_{\mathcal{P} \in \mathcal{MP}} \mathrm{PHS}(s,t \mid \mathcal{P}).$$

Notice that if we select a metapath from $\mathcal{MP}$ uniformly at random and took the HeteSim score relative to that metapath, the expected value of the score is precisely $Q(s, t)$. Thus, we may approximate $Q(s, t)$ by taking m independent and uniformly chosen math-paths, $\mathcal{P}_1, \cdots, \mathcal{P}_m$, and taking the mean of the HeteSim scores relative to these metapaths. Let

$$\widehat{Q}(s,t) = \frac{1}{m} \sum_{i=1}^{m} HS(s,t \mid \mathcal{P}_i).$$

Hence, $\mathbb{E}(\widetilde{R}(s,t)) = R(s,t)$.

Let $\widetilde{\mathrm{PHS}}(s,t \mid \mathcal{P})$ be the approximation of $\mathrm{PHS}(s,t \mid \mathcal{P})$ derived from our randomized algorithm after taking $n(s,t \mid \mathcal{P})$ random walks. Let $k(s,t \mid \mathcal{P})$ be the number of reachable nodes of type $A_{l/2+1}$ when considering source s, target t and metapath $\mathcal{P}$. Let $k_{\max} = \max\{k(s,t \mid \mathcal{P}_1), \cdots, k(s,t \mid \mathcal{P}_m)\}$, for $\mathcal{MP}_{ST} = \{\mathcal{P}_1, \ldots, \mathcal{P}_m\}$. By the construction of the algorithm, $\mathbb{E}(\widetilde{\mathrm{PHS}}(s,t \mid \mathcal{P})) = \mathrm{PHS}(s,t \mid \mathcal{P})$ for a fixed $\mathcal{P}$. Let

$$\tilde{R}(s,t) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{PHS}(s,t \mid \mathscr{P}_i)$$

and

$$\hat{R}(s,t) = \frac{1}{m}\sum_{i=1}^{m} \widehat{\mathrm{PHS}}(s,t \mid \mathscr{P}_i).$$

Similarly to the above, $\mathbb{E}(\tilde{R}(s,t)) = R(s,t)$. We now see that

$$\mathbb{E}(\hat{R}(s,t)) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left(\mathbb{E}\left(\widehat{\mathrm{PHS}}(s,t \mid \mathscr{P}_i) \mid \mathscr{P}_i\right)\right) = \mathbb{E}\left(\frac{1}{m}\sum_{i=1}^{m} \mathrm{PHS}(s,t \mid \mathscr{P}_i)\right) = \mathbb{E}(\tilde{R}(s,t)) = R(s,t).$$

We now provide bounds on the number of random metapaths (m) we require to have $\hat{Q}(s,t)$ and $\hat{R}(s,t)$ be within some error of $Q(s, t)$ and $R(s, t)$, respectively, with at least some probability.

Lemma 5 (Bounded differences inequality [41]). *Let $Z_1, \cdots, Z_k$ be independent random variables such that $Z_i \in \Lambda_i$. Let $f : \Lambda_1 \times \cdots \times \Lambda_k \to \mathbb{R}$. Assume there exist $c_1, \cdots, c_k \in \mathbb{R}$ such that, for all i,*

$$\left|f(a_1, \cdots, a_i - 1, a_i, a_i + 1, \cdots a_k) - f(a_1, \cdots, a_i - 1, a_i', a_i + 1, \cdots a_k)\right| \leq c_i$$

for all $a_j \in \Lambda_j$ and $a_i' \in \Lambda_i$. Let $X = f(Z_1, \cdots, Z_k)$. We have that

$$Pr(|X - \mathbb{E}(X)| \geq t) \leq 2\, exp\left(-\frac{2t^2}{\sum_{i=1}^{k} c_i^2}\right).$$

Lemma 6. *For all $(s, t) \in S \times T$,*

$$Pr\left(\left|\hat{Q}(s,t) - Q(s,t)\right| \geq \epsilon\right) \leq 2e^{-2m\epsilon^2}$$

and

$$Pr\left(\left|\tilde{R}(s,t) - R(s,t)\right| \geq \epsilon\right) \leq 2e^{-2m\epsilon^2}$$

for all $(s, t) \in S \times T$.

Proof. Fix $(s, t) \in S \times T$. We utilize the bounded differences inequality. We take $\mathscr{P}_1, \cdots, \mathscr{P}_m$ to be our independent random variables. Let

$$\widehat{Q}(\mathscr{P}_1, \cdots \mathscr{P}_k \cdots, \mathscr{P}_m)(s,t) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{HS}(s,t \mid \mathscr{P}_i).$$

Notice that for any $k \in [m]$,

$$\left|\widehat{Q}(\mathscr{P}_1, \cdots \mathscr{P}_k \cdots, \mathscr{P}_m)(s,t) - \widehat{Q}(\mathscr{P}_1, \cdots \mathscr{P}'_k \cdots, \mathscr{P}_m)(s,t)\right| = \left|\frac{\mathrm{HS}(s,t \mid \mathscr{P}_k) - \mathrm{HS}(s,t \mid \mathscr{P}'_k)}{m}\right|$$
$$\leq \frac{1}{m}.$$

Thus, $C_i = \frac{1}{m}$ is sufficient to apply the bounded differences inequality. Hence,

$$Pr\left(\left|\widehat{Q}(s,t) - \mathbb{E}(\widehat{Q}(s,t))\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{m} c_i^2}\right) = 2e^{-2m\epsilon^2}.$$

Similar argument holds for $\widetilde{R}(s,t)$. □

Corollary 3. *For* $m = \frac{1}{2\epsilon^2}\ln\left(\frac{2|S||T|}{r}\right)$, *with probability at least* $1 - r$,

$$\left|\widehat{Q}(s,t) - Q(s,t)\right| < \epsilon$$

for all $(s, t) \in S \times T$.

Proof. Applying Lemma 6, we see that

$$Pr\left(\bigcup_{(s,t) \in S \times T} \left|\widehat{Q}(s,t) - Q(s,t)\right| \geq \epsilon\right)$$
$$\leq \sum_{(s,t) \in S \times T} Pr(\mid \widehat{Q}(s,t) - Q(s,t)) \mid \geq \epsilon\Bigg)$$
$$\leq 2|S||T|e^{-2m\epsilon^2}.$$

Thus, the probability that $\left|\widetilde{R}(s,t) - R(s,t)\right| < \epsilon$ for all $(s, t) \in S \times T$ is at least $1 - 2|S||T|e^{-2m\epsilon^2}$. To have this probability at least $1 - r$, it is hence sufficient to have $2|S||T|e^{-2m\epsilon^2} = r$, proving the result. □

Theorem 3. *Fix* $0 < \epsilon, r < 1$. *For*

$$n(s,t \mid \mathscr{P}_i) = \frac{4c\left(\frac{\epsilon}{2}\right) \cdot k(s,t \mid \mathscr{P}_i)}{\epsilon^2} \ln\left(\frac{4m|S||T|k_{\max}}{r_1}\right)$$

and

$$m = \frac{2}{\epsilon^2}\ln\left(\frac{2|S||T|}{r - r_1}\right),$$

$$\textit{where } r_1 = r \cdot \frac{4\ln\left(\frac{2|S||T|}{r}\right)k_{\max}}{4\ln\left(\frac{2|S||T|}{r}\right)k_{\max} + \epsilon^2}, \textit{ with probability at least } 1 - r,$$

$$\left|\widehat{R}(s,t) - R(s,t)\right| < \epsilon$$

for all $(s, t) \in S \times T$.

The proof of this result is deferred to Appendix A.

The results from this section give rise to 2 algorithms for computing approximations to mean HeteSim scores. First, Corollary 3 gives an algorithm for approximating the mean HeteSim score using the deterministic HeteSim algorithm given in Algorithm 1. Pseudocode for this approximate mean HeteSim computation is given in Algorithm 7. Second, Theorem 3 shows how to compute an approximation to the mean pruned HeteSim score, and pseudocode for this computation is given in Algorithm 8.

Algorithm 7: Approximate Mean HeteSim score.

Input: set of start nodes S, end node t, path length p, approximation parameters ϵ and r
Output: vector of approximate mean HeteSim scores h, indexed by elements of S, with error bounds as in Corollary 3
$m \leftarrow \frac{1}{2\epsilon^2}\ln\left(\frac{2|S|}{r}\right)$
Construct M, the set of all metapaths of length p between any element of S and t
if $m < M$ **then**
  select $M' \subseteq M$ with $|M'| = m$ uniformly at random
else
  $M' \leftarrow M$
end if
for $s \in S$ **do**
  HSscores = []
  for $m \in M'$ **do**
    HSscores.append(HeteSim(s, t, m))
  end for
  $h[s]$ = mean(HSscores)
end for
return h

3.3. Algorithm Runtimes: SemNet Version 2

Having given algorithms and proofs of correctness, we now turn to a computational investigation of actual algorithm performance. Our emphasis is on comparing the three different algorithms enumerated above.

3.3.1. Verification of Randomized Algorithm Performance—For each of the three test graphs and corresponding metapaths, the randomized pruned HeteSim algorithm was run 100 times, with $\epsilon = 0.05$ and $r = 0.95$. For each of the three test graphs, an error less than ϵ was observed in all 100 iterations. Histograms showing the distribution of computed values are given in Figure 8.

Algorithm 8: Approximate Mean Pruned HeteSim score.

Input: set of start nodes S, end node t, path length p, approximation parameters ϵ and r

Output: vector of approximate mean HeteSim scores h, indexed by elements of S, with error bounds as in Theorem 3

$r_1 \leftarrow r \cdot \frac{4\ln\left(\frac{2|S||T|}{r}\right)k_{max}}{4\ln\left(\frac{2|S||T|}{r}\right)k_{max}+\epsilon^2}$

$m \leftarrow \frac{2}{\epsilon^2}\ln\left(\frac{2|S|}{r-r_1}\right)$

$N \leftarrow \frac{4c\left(\frac{r}{2}\right)\cdot k(s,t|P_l)}{\epsilon^2}\ln\left(\frac{4m|S||T|k_{max}}{r_1}\right)$

Construct M, the set of all metapaths of length p between any element of S and t

if $m < M$ **then**

  select $M' \subseteq M$ with $|M'| = m$ uniformly at random

else

  $M' \leftarrow M$

end if

for $s \in S$ **do**

  PHSscores = []

  for $m \in M'$ **do**

    PHSscores.append(RandomizedPrunedHeteSimGivenN(s, t, m, N))

  end for

  $h[s]$ = mean(PHSscores)

end for

return h

3.3.2. Comparison of Algorithm Runtimes—For two of the three main algorithm variants, runtime on length 2 metapaths was measured, using Alzheimer's disease as a target node and a set of three source nodes: insulin, hypothyroidism, and amyloid. Each of these source nodes has some amount of real-world domain significance; all three have, at some point, acted as a source node to the target node Alzheimer's disease in other ongoing research in the authors' lab. This ongoing work aims to investigate and discover causes and treatments (re-purposed or otherwise) within the active body of biomedical academic literature. As a more specific example, SemNet version 1 was used to investigate how hypothyroidism and Alzheimer's disease are related via the combined rankings of shared source nodes. This is a slightly different application than what is being investigated in this manuscript, but the results definitively show that hypothyroidism and Alzheimer's disease are closely related. These previous runs have historically been extremely slow while utilizing SemNet version 1, taking up to an hour to complete (see Table 2). Decreasing runtime is the main motivation for the new algorithms and implementations.

For the two main chosen algorithms associated with SemNet version 2, mean exact HeteSim and approximate mean HeteSim, test runs were conducted using the previously defined source–target combinations. These test runs were repeated 10 times per combination for both algorithms respectively, and the comprehensive runtime results can be seen in Table 3. For approximate mean HeteSim, the realistic parameters $\epsilon = 0.1$ and $r = 0.9$ were used. The third algorithm variant, approximate mean pruned HeteSim, was not run on the actual knowledge graph, due to excessive runtime when using realistic values for ϵ and r.

For the fastest algorithm, approximate mean HeteSim, time spent on each of the three steps described in Figure 5 was also recorded. To further accentuate the speed differences between SemNet versions 1 and 2 (specifically approximate mean HeteSim), Table 2 shows the three step breakdown for both SemNet versions side by side. For both versions, the same target (Alzheimer's disease) and sources (insulin, hypothyroidism, and amyloid) were used, and each source–target combination, like in Table 3, was run 10 times each. The runtime ratio between SemNet version 1 and SemNet version 2 is also shown in Table 2. For these three step breakdown tests, the approximate mean HeteSim algorithm used the parameters $\epsilon = 0.1$ and $r = 0.9$ once again.

Additionally, the time to compute HeteSim using the new data structure for a single metapath was analyzed. Due to the HeteSim algorithms being run on single metapaths, aggregation (Step 3) was not used and therefore not represented in timing results. For comparison, the top 20 unique metapaths, based on the metapaths with the highest number of unique paths (each metapath between a source and target node can potentially encompass many different paths), were used as inputs to the respective algorithms. Both the deterministic HeteSim and randomized pruned HeteSim algorithms were run on these metapaths, with approximation parameters $\epsilon = 0.1$ and $r = 0.9$ applied to the latter. Randomized pruned HeteSim was not run on all metapaths due to excessive runtime, and, therefore, deterministic HeteSim was also not run over all metapaths, for comparison sake. Results of this comparison are given in Table 4. Further detail on the randomized pruned HeteSim results, including the maximum and minimum values for the number of iterations, runtimes, and metapath instances (the number of paths within a metapath), is given in Table 5. Figure 9 shows the breakdown of deterministic HeteSim computation time for each metapath between the described sources and target, with no limit on the number of metapaths.

As a final timing comparison, the top 20 length 4 metapaths (again determined by the metapaths comprised of the highest number of unique paths) were generated for each of the three testing target-source node pairs, and the deterministic HeteSim algorithm was run on all 60 metapaths. The 20 length 4 metapaths were taken out of a subset of the first 100,000 total length 4 metapaths shared between each respective source node and AD. Metapath computation is the greatest bottleneck, and retrieving any more than 100,000 metapaths per source node is simply too time consuming as of right now. The maximum, minimum, and mean runtimes for this final test are shown in Table 6. As a final side note, different runs of both SemNet versions 1 and 2 might vary in computational time due to changes in concurrent computational load and random, extrinsic factors. This slight variation does not change the ultimate goal or conclusion of this study.

3.4. Study Assessing User Friendliness of SemNet Version 2

SemNet version SemNet version 1 had extensive Sphinx documentation and readme files, but there was no detailed example Jupyter interface for users with limited computer science or Python background to easily run the software. User friendliness was primarily assessed with a standardized survey of two distinct groups of naive or first-time SemNet software users who were trained in either SemNet version 1 or SemNet version 2. “Training” included a general introduction or background on the purpose and utility of the SemNet framework (same content for each group), along with publicly available user documentation (documentation to either SemNet version 1 or SemNet version 2, depending on user group assignment). The SemNet version 1 user group had 11 users (n = 11), whereas the SemNet version 2 user group had 10 users (n = 10). All participants were students at Georgia Institute of Technology.

To ensure that differences in prior experience with Python or Jupyter notebooks would not bias the user study results, each participant was asked to self-classify their prior experience using Python and/or Jupyter notebooks to ensure each user group had a balanced distribution

of prior Python/Jupyter user experiences. The Python experience classifications were: novice user (no to minimal Python experience); proficient user (had taken a basic Python class or had previously independently used Python for an elementary project); or expert Python user (very confident and capable of teaching a class on Python/Jupyter). The SemNet version 1 group included 3 participants who self-identified as novice Python users, 7 that self-identified as proficient Python users, and 1 that self-identified as an expert Python user. The SemNet version 2 group included 3 participants who self-identified as novice Python users, 6 that self-identified as proficient Python users, and 1 that self-identified as an expert Python user.

After completing a standardized training protocol, each user took an electronic survey asking a simple question: “Are you comfortable in running a [SemNet] simulation on your own?”. The SemNet version 1 group had 2 of 11 users who answered they were comfortable in running a SemNet version 1 simulation after minimal training. The SemNet version 2 group had 8 of 10 users that answered they were comfortable in running a SemNet version 2 simulation after minimal training. Fisher’s exact test compared these two user groups; the SemNet version 2 user group was significantly ($p < 0.05$) more comfortable performing a simulation compared to users in the SemNet version 1 group. This result quantitatively affirms that the SemNet version 2 framework is more user friendly and intuitive than SemNet version 1.

Finally, a random subset of users (n = 7) were eventually trained in both SemNet version 1 and SemNet version 2. These users were asked a simple question via an electronic survey: “Is the user friendliness of SemNet version 2 equal, somewhat better, or much better than SemNet version 1?” All 7 users said SemNet version 2 was “much better” than SemNet version 1. While the sample size is small, the probability that all 7 users select “much better” is significant ($p < 0.05$). SemNet version 2’s interface and greatly enhanced speed were the volunteered reasons stated for it being voted “much better” by users for its user friendliness.

3.5. Assessing Highly Ranked Metabolic Nodes to Alzheimer’s Disease

Recent literature has identified relationships shared between metabolic co-morbidities and AD [33,42,43]. The scope of the present article focuses on the mathematics, computational optimizations, performance improvements, and user friendliness of SemNet version 2. An entirely different manuscript could be dedicated to sifting through interesting results on the Alzheimer’s case study used to perform SemNet version 2 performance evaluations. Due to space constraints and article scope, we only briefly touch on some of the interesting nodes identified and ranked in SemNet version 2 using Alzheimer’s disease (AD) as the target node and hypothyroidism and insulin as source nodes of interest.

One of the key advantages of SemNet is examining multi-factorial relationships that are not as obvious. A small subset of lesser discussed source nodes involving metabolic co-morbidities and AD ranked as relatively important by SemNet version 2 include the following: metformin (a drug used to treat type 2 or adult-onset diabetes), dexamethasone (a glucocorticoid use to treat inflammation, autoimmune disease, or adrenal insufficiency), carbonic anhydrase (a family of enzymes that catalyze the interconversion between carbon dioxide and water), and nitric oxide synthase 3 (generates NO in blood vessels and is

involved with regulating vascular function). These specific source nodes are identified by finding all intersecting source nodes shared between AD and multiple targets (metabolic co-morbidities, in this example) and ranking all shared sources with respect to each AD-metabolic co-morbidity pairing. In this example, the chosen metabolic co-morbidities associated with AD are obesity, hypothyroidism, and type 2 diabetes [34,35,44]. The four example source node results mentioned above (metformin, dexamethasone, carbonic anhyrdrase, and nitric oxicde 3) scored very highly in each run of SemNet version 2, consistently placing in the top 25% of ranked nodes based on HeteSim score. More specific explanations for why or how these identified source nodes are tied to AD are discussed in studies contributing to the knowledge graph connectivity, some of which are cited here [45–48].

4. Discussion

The results presented in this manuscript show that the main objective, reducing SemNet's overall runtime, has been achieved. This increase in speed is attributable to both algorithmic improvements (best seen with the approximate mean HeteSim algorithm) and, most substantially, data structure changes. The secondary objective, fixing the error presented in the SemNet version 1 rank aggregation algorithm ULARA, was also met with the introduction of two new aggregation algorithms: exact mean aggregation and approximate mean aggregation. The success presented in this work will provoke a quick adoption of SemNet version 2. Computational challenges still remain, specifically in metapath enumeration and computation. The need to compute all metapaths between the specified source–target nodes is still a relatively major computational bottleneck to be addressed in future work.

4.1. Computational Improvements

Both the mean HeteSim score and approximate mean HeteSim score show runtime reductions compared to SemNet version 1. These improvements are evident both in the overall algorithm runtimes (Tables 1 and 3) and in the speed of the deterministic HeteSim subroutine (Tables 1 and 4). Note that, though the number of metapaths decreased in the graph used to test SemNet version 2 and this reduction must account for some speedup, computation time per metapath decreased. Table 2 shows that the largest improvement happened in step 2, likely because the implementation of step 2 in SemNet version 1 used many Neo4j queries. Since it has already been shown that Neo4j queries made up most of the runtime in SemNet version 1 (see Table 1), it is likely that the substitution of the Python dictionary-based data structure for the knowledge graph was the largest source of runtime reduction for step 2. Similarly, step 1 involves querying the knowledge graph, and the replacement of Neo4j with a custom dictionary-based data structure is likely the largest source of improvement here as well.

Step 3 is a bit different because the changes here were motivated by the replacement of a flawed rank aggregation technique, rather than runtime considerations. As a ratio, we do see an improved reduction in runtime of over 1000, but the absolute runtime values for step 3 are quite small in relation to the entire algorithm. The most important result regarding

step 3 is the replacement ULARA with a sensible alternative (mean HeteSim score) that is also is amenable to approximation based on randomization. In the length 2 metapath tests reported in Table 3, the approximate mean HeteSim algorithm achieves a 20% runtime reduction compared to the exact mean HeteSim score computation. This reduction is mostly attributable to the need to run the HeteSim subroutine on fewer metapaths. Since the bound on the number of metapaths for which HeteSim must be computed depends only on the number of candidate source nodes and the approximation parameters ϵ and r (see Corollary 3), the performance advantage of the approximate mean computation should be even more substantial in situations involving more metapaths. This performance advantage will only become more pronounced when running the approximate mean HeteSim algorithm on longer metapaths because, generally, the longer the metapath the greater the instances of that metapath within the graph. As a final note, the use of approximation algorithms, or more tangibly the tradeoff of some accuracy for a large performance boost, is appropriate in this context. This conclusion is drawn from two generalizations: the knowledge graph is inherently noisy, as it is generated using natural language processing techniques on biomedical paper abstracts, and the primary use of SemNet is in hypothesis generation. Both factors make the accuracy/speed tradeoff an allowable, and generally preferable, possibility that might not be available in different contexts.

4.2. Mathematical Limitations

In Corollary 2, we provide a bound that demonstrated that it is sufficient to make $O\left(\frac{1}{\epsilon^2}k\ \ln\left(\frac{k}{1-r}\right)\right)$ random walks in the randomized Pruned HeteSim algorithm. As illustrated by Table 5, the bound we achieved may, at times, result in a large number of required walks, when considering realistic knowledge graphs and modest values for ϵ and r. We acknowledged that the bound we achieved may be crude, especially in our frequent use of the, generally loose, union bound. Hence, we leave open the possibility of substantial improvement to both the constant we achieve ($c(\epsilon) \leq 71$) and the order with respect to the various variables.

One possible area of improvement is in the order with respect to k. We conjecture that the required number of walks is at least order k, thus leaving room for the possibility of the true value to be between order k and $k \log k$ (inclusive). Considering the order with respect to ϵ, we note that most standard general concentration inequalities necessitate $O\left(\frac{1}{\epsilon^2}\right)$. This being said, the distribution we are considering is binomial. While the authors are not aware of any stronger results for the binomial distribution, we are also not aware of any reason why such a result could not exist.

We also note that to achieve Lemma 4, we utilize an error allocation scheme that bounds large entries with error proportional to the value of the entry but bounds small entries with a fixed bound. This is just one possible scheme which leaves open the possibility of achieving tighter results using another, possibly more individualized, scheme.

4.3. Limitations and Future Directions

The knowledge graph used to test SemNet version 2 has substantially fewer edges than the knowledge graph used in SemNet version 1, as seen by the reduced number of metapaths between vertices of interest (see Tables 1 and 2). The new graph was built to reduce the number of overly generic edges and redundant conclusions occasionally seen in SemNet version 1; the new graph is, overall, both better performing and more useful for hypothesis generation compared to the old graph. Future work will address this limitation and give more accurate runtime comparisons by building a knowledge graph of comparable size to that used in SemNet version 1, though this endeavor would mostly just be a confirmatory effort to give more precise runtime improvements.

Though the new implementation has significantly reduced the runtime required to enumerate metapaths, metapath enumeration remains a computational bottleneck. This bottleneck is a barrier to HeteSim computations on longer metapaths; this work has made length 4 metapath analysis feasible, though anything greater is potentially still unattainable. Since counting the number of paths between two specified nodes in a directed graph is #**P**-complete [49], metapath enumeration is likely also a computationally hard problem. To make further progress, future work will need to address this metapath enumeration problem. One possible approach is to devise an algorithm for sampling metapaths under a uniform (or other useful) probability distribution, perhaps using a Markov chain Monte Carlo technique similar to the approach employed in [50]. If such an algorithm could be devised, it could be used directly with the randomized aggregation scheme described in Algorithm 7.

4.4. Related Work

In this section, SemNet version 2 (i.e., SemNet 2.0) is compared and contrasted to other existing automated LBD tools.

4.4.1. Biomedical Knowledge Graphs—While a number of companies boast commercial biomedical knowledge bases, most publicly available KBs are limited in scope and diversity of node types. Many of these are created by aggregating specific, high-quality databases together. Databases in this category include Hetio [51], a KG built for drug re-purposing containing 48 K, 2.2 M edges, and 22 node types; OGB-BioKG from Open Graph Benchmark [52], a general-purpose biomedical KG containing 93 K nodes, 5 M edges, 5 node types, and 51 edge types (most of which are specific drug-drug interactions); DRKG [53] is a drug repositioning knowledge graph for COVID-19 that combines entities/relations from 6 existing databases with additional entity and relationship data extracted from open-source biomedical literature on COVID-19.

Other biomedical KGs have been created by using natural language processing to extract information from biomedical text. PubMed Knowledge Graph [54], which creates a paper-centric knowledge graph by linking authors, entities, institutions, and funding sources to research articles and connecting articles via citations. SemMedDB [8] contains a approximately 100 M (subject, object, predicate) triples extracted from PubMed articles from 124 node types and 58 relation types, each of which is linked to the article from which it was taken. SemNet 2.0's knowledge graph is derived from a processed version of

SemMedDB which removes links to papers and aggregates relation triples to more directly identify the relationships between biological entities.

4.4.2. Related Algorithms—At its core, SemNet 2.0 is a framework for identifying relatedness among nodes in a knowledge graph. This is similar to other knowledge base completion (KBC) algorithms, which seek to identify missing edges between knowledge graph nodes. A large family of knowledge base completion algorithms seek to infer missing edges by modeling entity and relation representations as latent embeddings and learning these by encourage them to satisfy certain geometric properties. For example, TransE treats each entity as a point in Euclidean space and assumes that relations can be effectively modeled as translations between entity embeddings, i.e., $s + r \approx t$ for source node s, target node t, and relation r. A wide variety of other models operate on some variant of this assumption, substituting translation by element-wise scaling [55], rotation in complex space [56], rotation in Quaternary space [57], or rotation and reflection in hyperbolic space [58]. An smaller, alternative family of knowledge base completion literature focuses instead on inferring missing relations by aggregating information either explicitly [59] or implicitly [60,61] encoded in the (meta)paths between them. This approach is more desirable for biomedical KBs due to the fact that relevant nodes and paths can be extracted from the graph to provide an understandable explanation of the predictions. SemNet 2.0 is most similar to this family of path-based KBC models but differs in that SemNet 2.0 computes a general measure of relatedness instead of predicting the specific type of relation between KB entities.

5. Conclusions

In conclusion, with novel biomedical research constantly being generated and computational power ever increasing, literature-based discovery is here to stay. LBD is a field that will only become more relevant as time goes on, but for it to achieve user adoption at a large scale, tools and methods must be created that allow for efficient LBD to take place. SemNet, a tool that was first developed in 2019, is a novel attempt at performing LBD with an approach that, up to this point, has rarely been observed. SemNet departs from existing attempts by being both domain agnostic and simple to use, two features uncommon in current LBD systems. These features enable users of SemNet to quickly navigate the comprehensive biomedical concept graph and begin generating ranked lists of concepts that will ultimately facilitate new hypothesis generation. SemNet version 1 was the first iteration of SemNet, and it largely succeeded at being both an LBD tool and a general-purpose starting point for essentially any biomedical investigation that relies, in some capacity, on literature-based data. Through widespread, practical adoption, potential improvements for SemNet version 1 became apparent, particularly regarding runtime and HeteSim score aggregation for source nodes. SemNet version 2 (i.e., SemNet 2.0) addresses these problems in three predominant ways: an improved graph data structure, improved HeteSim implementations, and improved HeteSim score aggregation. With these advancements, SemNet 2.0 is a major step forward in improving the efficiency and efficacy of interactive automated LBD tools.

SemNet 2.0 has been compiled into a Python package. This package, along with the SemMedDB data required to build the biomedical concept graph, is open source. Detailed

documentation has been included with the package, all of which can be downloaded on GitHub.

Funding:

This research was funded by the National Science Foundation Graduate Research Fellowship Program to A.K., the McCamish Parkinson's Disease Innovation Program at Georgia Institute of Technology and Emory University to C.M., National Science Foundation CAREER grant 1944247 to C.M, National Institute of Health grant R21-CA232249 to C.M., Alzheimer's Association Research Grant Award 2018-AARGD-591014 to C.M., Goizueta Alzheimer's Disease Research Center at Emory University grant awards P50 AG025688 and P30 AG066511 to C.M., National Institute of Health grant U19-AG056169 sub-award to C.M. and Children's Hospital of Atlanta Aflac Pilot Grant Award to C.M.

Conflicts of Interest:

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Data Availability Statement:

SemNet 2.0 code can be found on GitHub https://github.com/pathology-dynamics/semnet-2 (accessed on 10 January 2022). Access instructions to SemRep/SemMedDB/SKR Resources for non-commercial use can be found at https://lhncbc.nlm.nih.gov/ii/tools/SemRep_SemMedDB_SKR.html (accessed on 10 January 2022).

Appendix A

Appendix A.1. Technical Lemmas

Lemma A1. *For* $v \in \mathcal{S}_k$, $\frac{1}{\sqrt{k}} \leq |v| \leq 1$.

Proof. By method of Lagrange Multipliers. □

Lemma A2. *Let* $\delta > 0$, $\alpha > 0$ *and* $0 < \beta \leq 1$. *Let* $v, w \in \mathcal{S}_k$ *and* $\lambda \in \mathcal{C}_k(v, \delta, \alpha, \beta)$. *We have that*

$$|\lambda \cdot v| \leq \delta\left(\left|v\right|^2 + \frac{k\beta^2}{4}\right) \quad \textit{and} \quad \left|\lambda \cdot \frac{w}{|w|}\right| \leq |\lambda| \leq \delta\sqrt{k\beta^2 + \left|v\right|^2}.$$

Proof. Assume v_I has m entries less than α and these are the first m entries. Clearly, $m \leq k$. We see that

$$\begin{aligned}
\lambda \cdot v &\leq \sum_{i=1}^{k} \lambda_i \cdot v_i \\
&\leq \sum_{i=1}^{m} \beta\delta \cdot v_i + \sum_{i=m+1}^{k} \delta \cdot v_i^2 \\
&\leq \delta \sum_{i=1}^{m} v_i(\beta - v_i) + \delta \sum_{i=1}^{k} \cdot v_i^2 \\
&\leq \delta\left|v\right|^2 + \delta\frac{k\beta^2}{4}.
\end{aligned}$$

We obtain the lower bound similarly. Clearly, $m \leq k$. Thus, we also see that

$$\begin{aligned}\left|\lambda \cdot \frac{w}{|w|}\right| &\leq \left|\left|\lambda\right| \cdot \frac{\lambda}{|\lambda|} \cdot \frac{w}{|w|}\right| \leq |\lambda| \\ &\leq \sqrt{\sum_{i=1}^{m} (\beta\delta)^2 + \delta^2 \sum_{i=m+1}^{k} v_i^2} \\ &\leq \delta\sqrt{k\beta^2 + \left|v\right|^2}.\end{aligned}$$

□

Lemma A3. *For* $v, \lambda \in \mathbb{R}^k$ *such that* $v, v + \lambda \in \mathcal{S}_k$,

$$\|v + \lambda| - |v\| \leq \frac{2|\lambda \cdot v| + |\lambda|^2}{|v| + \frac{1}{\sqrt{k}}}.$$

Proof. We first see that

$$\begin{aligned}\| v + \lambda| - |v|| &= \frac{\| v + \lambda| + |v|| \cdot \| v + \lambda| - |v||}{\| v + \lambda| + |v||} \\ &= \frac{\left|\left|v + \lambda\right|^2 - \left|v\right|^2\right|}{\| v + \lambda| + |v||} \\ &\leq \frac{\left|\left|v + \lambda\right|^2 - \left|v\right|^2\right|}{\left|v\right| + \frac{1}{\sqrt{k}}}.\end{aligned}$$

We now see that

$$\begin{aligned}\left|v + \lambda\right|^2 &= (v + \lambda) \cdot (v + \lambda) = v \cdot v + 2\lambda \cdot v + \lambda \cdot \lambda \\ &= \left|v\right|^2 + 2\lambda \cdot v + \left|\lambda\right|^2\end{aligned}$$

$$\left|v + \lambda\right|^2 - \left|v\right|^2 \leq 2|\lambda \cdot v| + \left|\lambda\right|^2.$$

Similarly, we get that $|v|^2 - |v + \lambda|^2 \leq 2|\lambda \cdot v| + |\lambda|^2$. The desired result then follows. □

Lemma A4. *Let* $\beta \leq 1$. *Let* $0 < \delta$ *and* $\alpha, \beta \geq 0$. *For* $v \in \mathcal{S}_k$ *and* $\lambda \in \mathcal{C}_k(v, \delta, \alpha, \beta)$,

$$\left|\frac{(v + \lambda) \cdot w}{|v + \lambda\|w|} - \frac{v \cdot w}{|v\|w|}\right| \leq \delta\left(2 + \frac{k\beta^2}{2|v|^2} + \sqrt{\frac{k}{|v|^2}\beta^2 + 1}\right) + \delta^2\left(\frac{k\beta^2}{|v|^2} + 1\right).$$

Proof. We see that

$$\begin{aligned}\frac{(v+\lambda)\cdot w}{|v+\lambda||w|}-\frac{v\cdot w}{|v||w|}&=\frac{(v+\lambda)\cdot w}{|v+\lambda||w|}-\frac{(v+\lambda)\cdot w}{|v||w|}-\frac{\lambda\cdot w}{|v||w|}\\&=\frac{(v+\lambda)\cdot w}{|w|}\left(\frac{1}{|v+\lambda|}-\frac{1}{|v|}\right)-\frac{\lambda\cdot w}{|v||w|}\\&=\frac{(v+\lambda)\cdot w}{|v+\lambda||w|}\cdot\frac{|v|-|v+\lambda|}{|v|}-\frac{1}{|v|}\cdot\left(\lambda\cdot\frac{w}{|w|}\right)\end{aligned}$$

$$\begin{aligned}\left|\frac{(v+\lambda)\cdot w}{|v+\lambda||w|}-\frac{v\cdot w}{|v||w|}\right|&\le\frac{1}{|v|}\left(\|v|-|v+\lambda\|+\left|\lambda\cdot\frac{w}{|w|}\right|\right)\\&\le\frac{2\delta\left(\left|v\right|^2+\frac{k\beta^2}{4}\right)+\delta^2\left(k\beta^2+\left|v\right|^2\right)}{\left|v\right|^2+\frac{|v|}{\sqrt{k}}}+\delta\sqrt{\frac{k\beta^2}{|v|^2}+1}\\&\le\frac{\delta\left(2+\frac{k\beta^2}{2|v|^2}\right)+\delta^2\left(\frac{k\beta^2}{|v|^2}+1\right)}{1+\frac{1}{|v|\sqrt{k}}}+\delta\sqrt{\frac{k\beta^2}{|v|^2}+1}\,.\end{aligned}$$

The above inequality follows from $\left|\frac{v\cdot w}{|w||v|}\right|\le 1$, Lemmas A2 and A3. □

Lemma A5. *Fix* ϵ. *Let* $\beta\le 1$ *and* $\alpha\ge 0$. *For* $v\in\mathcal{S}_k$ *and* $\lambda\in\mathcal{C}_k(v,\alpha,\delta,\delta')$. *Let*

$$b=\frac{2+\frac{k\beta^2}{2|v|^2}}{1+\frac{1}{|v|\sqrt{k}}}+\sqrt{\frac{k\beta^2}{|v|^2}+1}\quad\text{and}\quad a=\frac{\frac{k\beta^2}{|v|^2}+1}{1+\frac{1}{|v|\sqrt{k}}}.$$

For $\delta\le\frac{\epsilon}{b+\sqrt{b^2+2a\epsilon}}$,

$$\left|\frac{(v+\lambda)\cdot w}{|v+\lambda||w|}-\frac{v\cdot w}{|v||w|}\right|\le\frac{\epsilon}{2}.$$

Proof. The result follows from Lemma A4. □

Appendix A.2. Proofs and Theorems

Proof. For ϵ_1, $\epsilon_2>0$ such that $\epsilon_1+\epsilon_2=\epsilon$, we see that

$$Pr\left(\left|\widehat{R}(s,t)-R(s,t)\right|\ge\epsilon\right)\le Pr\left(\left|\widehat{R}(s,t)-\widetilde{R}(s,t)\right|+\left|\widetilde{R}(s,t)-R(s,t)\right|\ge\epsilon\right)$$

$$\begin{aligned}&Pr\left(\left|\widehat{R}(s,t)-R(s,t)\right|\le\epsilon\right)\ge Pr\left(\left|\widehat{R}(s,t)-\widetilde{R}(s,t)\right|+\left|\widetilde{R}(s,t)-R(s,t)\right|\le\epsilon\right)\\&\ge Pr\left(\left|\widehat{R}(s,t)-\widetilde{R}(s,t)\right|\le\epsilon_1\cap\left|\widetilde{R}(s,t)-R(s,t)\right|\le\epsilon_2\right)\end{aligned}$$

$$Pr\left(\left|\widehat{R}(s,t) - R(s,t)\right| \geq \epsilon\right) \leq Pr\left(\left|\widehat{R}(s,t) - \widetilde{R}(s,t)\right| \geq \epsilon_1 \cup \left|\widetilde{R}(s,t) - R(s,t)\right| \geq \epsilon_2\right)$$
$$\leq Pr\left(\left|\widehat{R}(s,t) - \widetilde{R}(s,t)\right| \geq \epsilon_1\right) + Pr\left(\left|\widetilde{R}(s,t) - R(s,t)\right| \geq \epsilon_2\right).$$

Recall from Lemma 6 that

$$Pr\left(\left|R(s,t) - \widetilde{R}(s,t)\right| \geq \epsilon_2\right) \leq 2\exp\left(-2m \cdot \epsilon_2^2\right).$$

Furthermore, from Lemma 4,

$$Pr\left(\left|\widehat{R}(s,t) - \widetilde{R}(s,t)\right| \geq \epsilon_1\right) = Pr\left(\left|\frac{1}{m}\sum_{i=1}^{m} \mathrm{PHS}(s,t \mid \mathscr{P}_i) - \frac{1}{m}\sum_{i=1}^{m} \widetilde{\mathrm{PHS}}(s,t \mid \mathscr{P}_i)\right| \geq \epsilon_1\right)$$
$$\leq Pr\left(\frac{1}{m}\sum_{i=1}^{m}\left|\mathrm{PHS}(s,t \mid \mathscr{P}_i) - \widetilde{\mathrm{PHS}}(s,t \mid \mathscr{P}_i)\right| \geq \epsilon_1\right)$$

$$Pr\left(\left|\widehat{R}(s,t) - \widetilde{R}(s,t)\right| \leq \epsilon_1\right) \geq Pr\left(\frac{1}{m}\sum_{i=1}^{m}\left|\mathrm{PHS}(s,t \mid \mathscr{P}_i) - \widetilde{\mathrm{PHS}}(s,t \mid \mathscr{P}_i)\right| \leq \epsilon_1\right)$$
$$\geq Pr\left(\bigcap_{i=1}^{m}\left|\mathrm{PHS}(s,t \mid \mathscr{P}_i) - \widetilde{\mathrm{PHS}}(s,t \mid \mathscr{P}_i)\right| \leq \epsilon_1\right)$$

$$Pr\left(\left|\widehat{R}(s,t) - \widetilde{R}(s,t)\right| \geq \epsilon_1\right) \leq Pr\left(\bigcup_{i=1}^{m}\left|\mathrm{PHS}(s,t \mid \mathscr{P}_i) - \widetilde{\mathrm{PHS}}(s,t \mid \mathscr{P}_i)\right| \geq \epsilon_1\right)$$
$$\leq \sum_{i=1}^{m} Pr\left(\left|\mathrm{PHS}(s,t \mid \mathscr{P}_i) - \widetilde{\mathrm{PHS}}(s,t \mid \mathscr{P}_i)\right| \geq \epsilon_1\right)$$
$$\leq \sum_{i=1}^{m} 4k(s,t \mid \mathscr{P}_i)\exp\left(-\frac{n(s,t \mid \mathscr{P}_i)}{k(s,t \mid \mathscr{P}_i)} \cdot \frac{\epsilon_1^2}{c(\epsilon_1)}\right).$$

Hence, for all $(s, t) \in S \times T$,

$$Pr\left(\left|\widehat{R}(s,t) - R(s,t)\right| \geq \epsilon\right) \leq 2\exp\left(-2m\epsilon_2^2\right) + \sum_{i=1}^{m} 4k(s,t \mid \mathscr{P}_i)\exp\left(-\frac{n(s,t \mid \mathscr{P}_i)}{k(s,t \mid \mathscr{P}_i)} \cdot \frac{\epsilon_1^2}{c(\epsilon_1)}\right)$$

$$Pr\left(\bigcup_{(s,t) \in S \times T}\left|\widehat{R}(s,t) - R(s,t)\right| \geq \epsilon\right) \leq \sum_{(s,t) \in S \times T} Pr\left(\left|\widehat{R}(s,t) - R(s,t)\right| \geq \epsilon\right)$$
$$\leq 2|S||T|\exp\left(-2m \cdot \epsilon_2^2\right)$$
$$+ \sum_{(s,t) \in S \times T}\sum_{i=1}^{m} 4k(s,t \mid \mathscr{P}_i)\exp\left(-\frac{n(s,t \mid \mathscr{P}_i)}{k(s,t \mid \mathscr{P}_i)} \cdot \frac{\epsilon_1^2}{c(\epsilon_1)}\right).$$

Fix $r_1, r_2 > 0$ such that $r_1 + r_2 = r$. We now see that for

$$
\begin{aligned}
&n(s,t \mid \mathscr{P}_i) \\
&= \frac{c(\epsilon_1)\cdot k(s,t \mid \mathscr{P}_i)}{\epsilon_1^2}\ln\left(\frac{1}{r_1}\sum_{(s,t)\in S\times T}\sum_{i=1}^{m} 4k(s,t \mid \mathscr{P}_i)\right) \\
&\leq \frac{c(\epsilon_1)\cdot k(s,t \mid \mathscr{P}_i)}{\epsilon_1^2}\ln\left(\frac{4m|S||T|k_{\max}}{r_1}\right)
\end{aligned}
$$

and

$$
m = \frac{1}{2\epsilon_2^2}\ln\left(\frac{2|S||T|}{r_2}\right),
$$

we have

$$
Pr\left(\bigcup_{(s,t)\in S\times T}\left|R(s,t)-\tilde{R}(s,t)\right|\geq\epsilon\right)\leq r\,.
$$

We now notice that the total number of walks taken to run the algorithm (ignoring dead ends) is at most $m\cdot\max\{n(s,t \mid \mathscr{P}_i)\} = mn$, where $n = \frac{c(\epsilon_1)\cdot k_{\max}}{\epsilon_1^2}\ln\left(\frac{4m|S||T|k_{\max}}{r_1}\right)$. We optimize to minimize nm by setting $\epsilon_1 = \frac{\epsilon}{2}$ and $r_1 = r\cdot\frac{2m_0k_{\max}}{2m_0k_{\max}+1}$, where $m_0 = \frac{1}{2\epsilon_2^2}\ln\left(\frac{2|S||T|}{r}\right)$ is some approximation for m. (We do not claim that these choices are optimal.) □

Appendix B

Appendix B.1. Analysis of Just-in-Time (JIT) Dead-End Removal

In our given Pruned HeteSim algorithm, whenever a dead-end node is found, it is removed from the graph for all future walks. We model this as follows. Assume there are $m\in\mathbb{N}$ dead-end nodes. Let $w\in\mathbb{R}_{\geq 0}$ be the maximum probability of reaching any single dead end. Thus, the probability of reaching a dead end is at most mw. Let $\alpha\in\mathbb{R}_{\geq 0}$ be the probability of any given walk not ending in a dead end and let $\beta = mw+\alpha$.

We now analyse the number of non-dead-end walks we expect to take by the time we hit some fixed number of dead ends and the number of dead ends we expect to take by the time we hit some fixed number of non-dead ends.

In the JIT algorithm, whenever we hit a dead end, the probability of hitting a dead end in the future is affect as follows. Let $X_1,\cdots\in\{0,1\}$, where $X_i = 1$ if the ith walk is not a dead end and $X_i = 0$ otherwise. For all i,

$$
Pr(X_i = 1) = \frac{\alpha}{\beta - wY_i'},
$$

where Y_i is the number of $X_j = 0$ for $j < i$. (Thus, treating w as the weight of each dead end and α as the weight on non-dead ends, each time we hit a dead end, the weight of the dead end hit is lost as we can no longer get to that dead end. This means that overtime the probability of hitting a dead end decreases.)

Let S_i be the number of $X_j = 1$ before the i-th $X_j = 0$. Let T_i be the number of $X_j = 0$ before the i-th $X_j = 1$.

Theorem A1. *For all* $i \in \mathbb{N}$,

$$S_i = \sum_{j=0}^{i-1} Z_j - i,$$

where Z_j *is geometrically distributed with parameter* $\frac{(m-j)w}{\beta - wj}$.

Proof. Notice that after the kth $X_j = 0$, the probability of $X_j = 1$ is $\frac{(m-k)w}{\beta - wk}$ (the probability of $X_j = 0$ after that point). Between the k-th $X_j = 0$ and $(k+1)$-th $X_j = 0$, this probability is fixed. Thus, the number of $X_j = 1$ between the k-th $X_j = 0$ and $(k+1)$-th $X_j = 0$ is geometrically distributed with parameter $\frac{(m-k)w}{\beta - wk}$. (We subtract i to not count the $X_j = 0$). □

Theorem A2.

$$\mathbb{E}(S_i) = \frac{\alpha}{w} \sum_{j=0}^{i-1} \frac{1}{m-j}$$

and

$$Var(S_i) = \frac{\alpha}{w^2} \sum_{j=0}^{i-1} \frac{\alpha + w(m-j)}{(m-j)^2}.$$

Proof. Follows from linearity of expectation and standard results about the geometric distribution. □

Remark A1. *We can obtain a bound of the deviation from the mean using Chebyshev's inequality.*

Lemma A6. *For* $i \geq 2$ *and* $k_{i-1} \leq k_i \leq m$,

$$Pr(T_i = k_i \mid T_{i-1} = k_{i-1}) = \frac{\alpha}{\beta - wk_i} \cdot \prod_{t=k_{i-1}}^{k_i - 1} \frac{(m-t)w}{\beta - tw}$$

and

$$Pr(T_1 = k_1) = \frac{\alpha}{\beta - wk_1} \cdot \prod_{t=0}^{k_1 - 1} \frac{(m-t)w}{\beta - tw}.$$

Proof. The distribution is similar to a geometric distribution with the change that for each failure the probability of success changes. □

We now see that

$$Pr(T_n = k_n, \cdots, T_1 = k_1) = Pr(T_1 = k_1) \prod_{i=2}^{n} Pr(T_i = k_i \mid T_i - 1 = k_i - 1)$$
$$= \alpha^n \left(\prod_{i=1}^{n} \frac{1}{\beta - wk_i} \right) \cdot \left(\prod_{t=0}^{k_n - 1} \frac{(m-t)w}{\beta - tw} \right)$$

$$Pr(T_n = k_n) = \left(\frac{\alpha}{\beta}\right)^n \left(\prod_{t=0}^{k_n - 1} \frac{(m-t)w}{\beta - tw} \right) \sum_{k_n - 1 = 0}^{k_n} \cdots \sum_{k_1 = 0}^{k_2} \left(\prod_{i=1}^{n} \frac{1}{1 - \frac{w}{\beta} k_i} \right).$$

Lemma A7. *For all* $n \geq 1$,

$$\sum_{k_n - 1 = 0}^{k_n} \cdots \sum_{k_1 = 0}^{k_2} \left(\prod_{i=1}^{n} \frac{1}{1 - \frac{w}{\beta} k_i} \right) \leq \frac{(-1)^{n-1}}{(n-1)!} \left(\frac{\beta}{w}\right)^{n-1} \frac{\ln\left(1 - \frac{w}{\beta}(k_n + n - 1)\right)^{n-1}}{1 - \frac{w}{\beta} k_n}.$$

Proof. We note that for increasing f,

$$\int_{a-1}^{b} f(x)\, dx \leq \sum_{k=a}^{b} f(k) \leq \int_{a}^{b+1} f(x) dx.$$

We prove this result inductively. Assume the result holds for $n = m$. For the upper bound, we now see that

$$\sum_{k_n - 1 = 0}^{k_{m+1}} \cdots \sum_{k_1 = 0}^{k_2} \left(\prod_{i=1}^{m+1} \frac{1}{1 - \frac{w}{\beta} k_i} \right)$$
$$= \frac{1}{1 - \frac{w}{\beta} k_{m+1}} \sum_{k_m = 0}^{k_{m+1}} \left(\sum_{k_m - 1 = 0}^{k_m} \cdots \sum_{k_1 = 0}^{k_2} \left(\prod_{i=1}^{m} \frac{1}{1 - \frac{w}{\beta} k_i} \right) \right)$$
$$\leq \frac{1}{1 - \frac{w}{\beta} k_{m+1}} \cdot \frac{1}{(m-1)!} \left(\frac{\beta}{w}\right)^{m-1} \sum_{k_m = 0}^{k_{m+1}} \frac{(-1)^{m-1} \ln\left(1 - \frac{w}{\beta}(k_m + m - 1)\right)^{m-1}}{1 - \frac{w}{\beta} k_m}.$$

We note that

$$\frac{\left[-\ln\left(1-\frac{w}{\beta}(k_m+m-1)\right)\right]^{m-1}}{1-\frac{w}{\beta}k_m}$$

is increasing as a function in k_m and positive for $k_m \geq 0$ for all $m \in \mathbb{N}$. Hence,

$$\begin{aligned}
&\sum_{k_n-1=0}^{k_m+1} \cdots \sum_{k_1=0}^{k_2} \left(\prod_{i=1}^{m+1} \frac{1}{1-\frac{w}{\beta}k_i} \right) \\
&\leq \frac{1}{1-\frac{w}{\beta}k_{m+1}} \cdot \frac{(-1)^{m-1}}{(m-1)!} \left(\frac{\beta}{w}\right)^{m-1} \int_0^{k_{m+1}+1} \frac{\ln\left(1-\frac{w}{\beta}(x+m-1)\right)^{m-1}}{1-\frac{w}{\beta}x} dx \\
&\leq \frac{1}{1-\frac{w}{\beta}k_{m+1}} \cdot \frac{(-1)^{m-1}}{(m-1)!} \left(\frac{\beta}{w}\right)^{m-1} \int_0^{k_{m+1}+1} \frac{\ln\left(1-\frac{w}{\beta}(x+m-1)\right)^{m-1}}{1-\frac{w}{\beta}(x+m-1)} dx \\
&\leq \frac{(-1)^m}{m!} \left(\frac{\beta}{w}\right)^m \frac{\left[\ln\left(1-\frac{w}{\beta}(k_{m+1}+m)\right)^m - \ln\left(1-\frac{w}{\beta}(m-1)\right)^m\right]}{1-\frac{w}{\beta}k_{m+1}} \\
&\leq \frac{(-1)^m}{m!} \left(\frac{\beta}{w}\right)^m \frac{\ln\left(1-\frac{w}{\beta}(k_{m+1}+m)\right)^m}{1-\frac{w}{\beta}k_{m+1}}.
\end{aligned}$$

□

Theorem A3.

$$Pr(T_n = k_n) \leq \left(\frac{\alpha}{w}\right)^n \left(\prod_{t=0}^{k_n-1} \frac{(m-t)w}{\beta - tw} \right) \frac{(-1)^{n-1}}{(n-1)!} \cdot \frac{\ln\left(1-\frac{w}{\beta}(k_n+n-1)\right)^{n-1}}{\frac{\beta}{w}-k_n}.$$

Proof. Follows from Lemma A7. □

References

1. PubMed Overview. Available online: https://pubmed.ncbi.nlm.nih.gov/about/ (accessed on 10 November 2021).
2. Swanson D Fish oil, Raynaud's syndrome, and undiscovered public knowledge. Perspect. Biol. Med 1986, 30, 7–18. [PubMed: 3797213]
3. Henry S; Wijesinghe DS; Myers A; McInnes BT Using Literature Based Discovery to Gain Insights Into the Metabolomic Processes of Cardiac Arrest. Front. Res. Metr. Anal 2021, 6, 32.
4. McCoy K; Gudapati S; He L; Horlander E; Kartchner D; Kulkarni S; Mehra N; Prakash J; Thenot H; Vanga SV; et al. Biomedical Text Link Prediction for Drug Discovery: A Case Study with COVID-19. Pharmaceutics 2021, 13, 794. [PubMed: 34073456]
5. Cameron D; Kavuluru R; Rindflesch TC; Sheth AP; Thirunarayan K; Bodenreider O Context-driven automatic subgraph creation for literature-based discovery. J. Biomed. Inform 2015, 54, 141–157. [PubMed: 25661592]

6. Crichton G; Baker S; Guo Y; Korhonen A Neural networks for open and closed Literature-based Discovery. PLoS ONE 2020, 15, e0232891. [PubMed: 32413059]
7. Sang S; Yang Z; Wang L; Liu X; Lin H; Wang J SemaTyP: A knowledge graph based literature mining method for drug discovery. BMC Bioinform. 2020, 19, 193.
8. Kilicoglu H; Shin D; Fiszman M; Rosemblat G; Rindflesch T SemMedDB: A PubMed-scale repository of biomedical semantic predications. Bioinformatics 2012, 28, 3158–3160. [PubMed: 23044550]
9. Himmelstein D; Lizee A; Hessler C; Brueggeman L; Chen S; Hadley D; Green A; Khankhanian P; Baranzini S Systematic integration of biomedical knowledge prioritizes drugs for repurposing. eLife 2017, 6, e26726. [PubMed: 28936969]
10. Li Y; Shi C; Yu PS; Chen Q HRank: A Path based Ranking Framework in Heterogeneous Information Network. In Web-Age Information Management; Springer International Publishing: New York, NY, USA, 2014; pp. 553–565.
11. Ng MK; Li X; Ye Y MultiRank: Co-ranking for objects and relations in multi-relational data. In Proceedings of the Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; pp. 1217–1225.
12. Shi C; Kong X; Huang Y; Yu PS; Wu B HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks. IEEE Trans. Knowl. Data Eng 2014, 26, 2479–2492.
13. Sedler AR; Mitchell CS SemNet: Using Local Features to Navigate the Biomedical Concept Graph. Front. Bioeng. Biotechnol 2019, 7, 156. [PubMed: 31334227]
14. Klementiev A; Roth D; Small K An Unsupervised Learning Algorithm for Rank Aggregation. In Machine Learning: ECML 2007; Kok JN, Koronacki J, Mantaras RLD, Matwin S, Mladenič D, Skowron A, Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 616–623.
15. Zeng X; Liao Y; Liu Y; Zou Q Prediction and Validation of Disease Genes Using HeteSim Scores. IEEE/ACM Trans. Comput. Biol. Bioinform 2017, 14, 687–695. [PubMed: 26890920]
16. Xiao Y; Zhang J; Deng L Prediction of lncRNA-protein interactions using HeteSim scores based on heterogeneous networks. Sci. Rep 2017, 7, 3664. [PubMed: 28623317]
17. Qu J; Chen X; Sun YZ; Zhao Y; Cai SB; Ming Z; You ZH; Li JQ In Silico Prediction of Small Molecule-miRNA Associations Based on the HeteSim Algorithm. Mol. Ther. Nucleic Acids 2019, 14, 274–286. [PubMed: 30654189]
18. Chen X; Shi W; Deng L Prediction of Disease Comorbidity Using HeteSim Scores based on Multiple Heterogeneous Networks. Curr. Gene Ther 2019, 19, 232–241. [PubMed: 31530261]
19. Fan C; Lei X; Guo L; Zhang A Predicting the Associations Between Microbes and Diseases by Integrating Multiple Data Sources and Path-based HeteSim Scores. Neurocomputing 2019, 323, 76–85.
20. Wang J; Kuang Z; Ma Z; Han G GBDTL2E: Predicting lncRNA-EF Associations Using Diffusion and HeteSim Features Based on a Heterogeneous Network. Front. Genet 2020, 11, 272. [PubMed: 32351537]
21. Garey MR; Graham RL; Ullman JD An Analysis of Some Packing Algorithms. Available online: https://mathweb.ucsd.edu/~ronspubs/73_08_packing.pdf (accessed on 10 January 2022).
22. Johnson DS Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci 1974, 9, 256–278.
23. Du DZ; Ko KI; Hu X Design and Analysis of Approximation Algorithms; Springer Science & Business Media: New York, NY, USA, 2011; Volume 62.
24. Vazirani VV Approximation Algorithms; Springer Science & Business Media: New York, NY, USA, 2013.
25. Williamson DP; Shmoys DB The Design of Approximation Algorithms; Cambridge University Press: Cambridge, UK, 2011.
26. What is a Graph Database? Available online: https://neo4j.com/developer/graph-database/#:~:text=Neo4j%20is%20an%20open%2Dsource,been%20publicly%20available%20since%202007 (accessed on 10 January 2022).
27. Weller J; Budson A Current understanding of Alzheimer's disease diagnosis and treatment. F1000Research 2018, 7, 1–9.

28. Thakur N; Han CY An Ambient Intelligence-Based Human Behavior Monitoring Framework for Ubiquitous Environments. Information 2021, 12, 81.
29. Hakansson K; Rovio S; Helkala EL; Vilska AR; Winblad B; Soininen H; Nissinen A; Mohammed AH; Kivipelto M Association between mid-life marital status and cognitive function in later life: Population based cohort study. BMJ 2009, 339.
30. Silva MVF; Loures CDMG; Alves LCV; de Souza LC; Borges KBG; Carvalho MDG Alzheimer's disease: Risk factors and potentially protective measures. J. Biomed. Sci 2019, 26, 33. [PubMed: 31072403]
31. Prakash J; Wang V; Quinn RE; Mitchell CS Unsupervised Machine Learning to Identify Separable Clinical Alzheimer's Disease Sub-Populations. Brain Sci. 2021, 11, 977. [PubMed: 34439596]
32. Huber CM; Yee C; May T; Dhanala A; Mitchell CS Cognitive decline in preclinical Alzheimer's disease: Amyloid-beta versus tauopathy. J. Alzheimer's Dis 2018, 61, 265–281. [PubMed: 29154274]
33. Johnson E; Dammer E; Duong D; Ping L; Zhou M; Yin L; Higginbotham L; Guajardo A; White B; Troncoso J; et al. Large-scale proteomic analysis of Alzheimer's disease brain and cerebrospinal fluid reveals early changes in energy metabolism associated with microglia and astrocyte activation. Nat. Med 2020, 26, 769–780. [PubMed: 32284590]
34. O'Barr SA; Oh JS; Ma C; Brent GA; Schultz JJ Thyroid hormone regulates endogenous amyloid-beta precursor protein gene expression and processing in both in vitro and in vivo models. Thyroid 2006, 16, 1207–1213. [PubMed: 17199430]
35. Matsuzaki T; Sasaki K; Tanizaki Y; Hata J; Fujimi K; Matsui Y; Sekita A; Suzuki S; Kanba S; Kiyohara Y; et al. Insulin resistance is associated with the pathology of Alzheimer disease. Neurology 2010, 75, 764–770. [PubMed: 20739649]
36. TPS Foundation Time. Available online: https://docs.python.org/3/library/time.html (accessed on 10 January 2022).
37. Gorelick M; Ozsvald I High Performance Python: Practical Performant Programming for Humans; O'Reilly Media: Sebastopol, CA, USA, 2020.
38. Jupyter P Jupyter Notebook. Available online: https://jupyter.org/ (accessed on 10 January 2022).
39. TPS Foundation Python. Available online: https://www.python.org/ (accessed on 10 January 2022).
40. Alon N; Spencer JH The Probabilistic Method; John Wiley & Sons: Hoboken, NJ, USA, 2004.
41. McDiarmid C On the method of bounded differences. Surv. Comb 1989, 141, 148–188.
42. Liu Q; Zhang J Lipid metabolism in Alzheimer's disease. Neurosci. Bull 2014, 30, 331–345. [PubMed: 24733655]
43. Chen Z; Zhong C Decoding Alzheimer's disease from perturbed cerebral glucose metabolism: Implications for diagnostic and therapeutic strategies. Prog. Neurobiol 2013, 108, 21–43. [PubMed: 23850509]
44. Alford S; Patel D; Perakakis N; Mantzoros CS Obesity as a risk factor for Alzheimer's disease: Weighing the evidence. Obes. Rev 2018, 19, 269–280. [PubMed: 29024348]
45. Li J; Deng J; Sheng W; Zuo Z Metformin attenuates Alzheimer's disease-like neuropathology in obese, leptin-resistant mice. Pharmacol. Biochem. Behav 2012, 101, 564–574. [PubMed: 22425595]
46. Hui Z; Zhijun Y; Yushan Y; Liping C; Yiying Z; Difan Z; Chunglit CT; Wei C The combination of acyclovir and dexamethasone protects against Alzheimer's disease-related cognitive impairments in mice. Psychopharmacology 2020, 237, 1851–1860. [PubMed: 32221697]
47. Sun MK; Alkon DL Carbonic anhydrase gating of attention: Memory therapy and enhancement. Trends Pharmacol. Sci 2002, 23, 83–89. [PubMed: 11830265]
48. Liu S; Zeng F; Wang C; Chen Z; Zhao B; Li K Carbonic anhydrase gating of attention: Memory therapy and enhancement. Sci. Rep 2015, 5.
49. Valiant LG The Complexity of Enumeration and Reliability Problems. SIAM J. Comput 1979, 8, 410–421.
50. Saha TK; Hasan MA Finding Network Motifs Using MCMC Sampling. In Complex Networks VI; Springer International Publishing: New York, NY, USA, 2015; pp. 13–24.

51. Himmelstein D; Baranzini S Heterogeneous Network Edge Prediction: A Data Integration Approach to Prioritize DiseaseAssociated Genes. PLoS Comput. Biol 2015, 11, e1004259. [PubMed: 26158728]
52. Hu W; Fey M; Zitnik M; Dong Y; Ren H; Liu B; Catasta M; Leskovec J Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv 2020, arXiv:2005.00687.
53. Ioannidis VN; Song X; Manchanda S; Li M; Pan X; Zheng D; Ning X; Zeng X; Karypis G DRKG —Drug Repurposing Knowledge Graph for COVID-19. 2020 Available online: https://github.com/gnn4dr/DRKG/ (accessed on 10 January 2022).
54. Xu J; Kim S; Song M; Jeong M; Kim D; Kang J; Rousseau JF; Li X; Xu W; Torvik VI; et al. Building a PubMed knowledge graph. Sci. Data 2020, 7, 205. [PubMed: 32591513]
55. Yang B; tau Yih W; He X; Gao J; Deng L Embedding Entities and Relations for Learning and Inference in Knowledge Bases. arXiv 2014, arXiv:1412.6575.
56. Sun Z; Deng ZH; Nie JY; Tang J RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
57. Zhang S; Tay Y; Yao L; Liu Q Quaternion Knowledge Graph Embeddings. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
58. Chami I; Wolf A; Juan DC; Sala F; Ravi S; Ré C Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 6901–6914.
59. Das R; Godbole A; Monath N; Zaheer M; McCallum A Probabilistic Case-based Reasoning for Open-World Knowledge Graph Completion. In Findings of the Association for Computational Linguistics: EMNLP 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 4752–4765.
60. Wang H; Ren H; Leskovec J Relational Message Passing for Knowledge Graph Completion. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 1697–1707.
61. Hu Z; Dong Y; Wang K; Sun Y Heterogeneous Graph Transformer. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2704–2710.

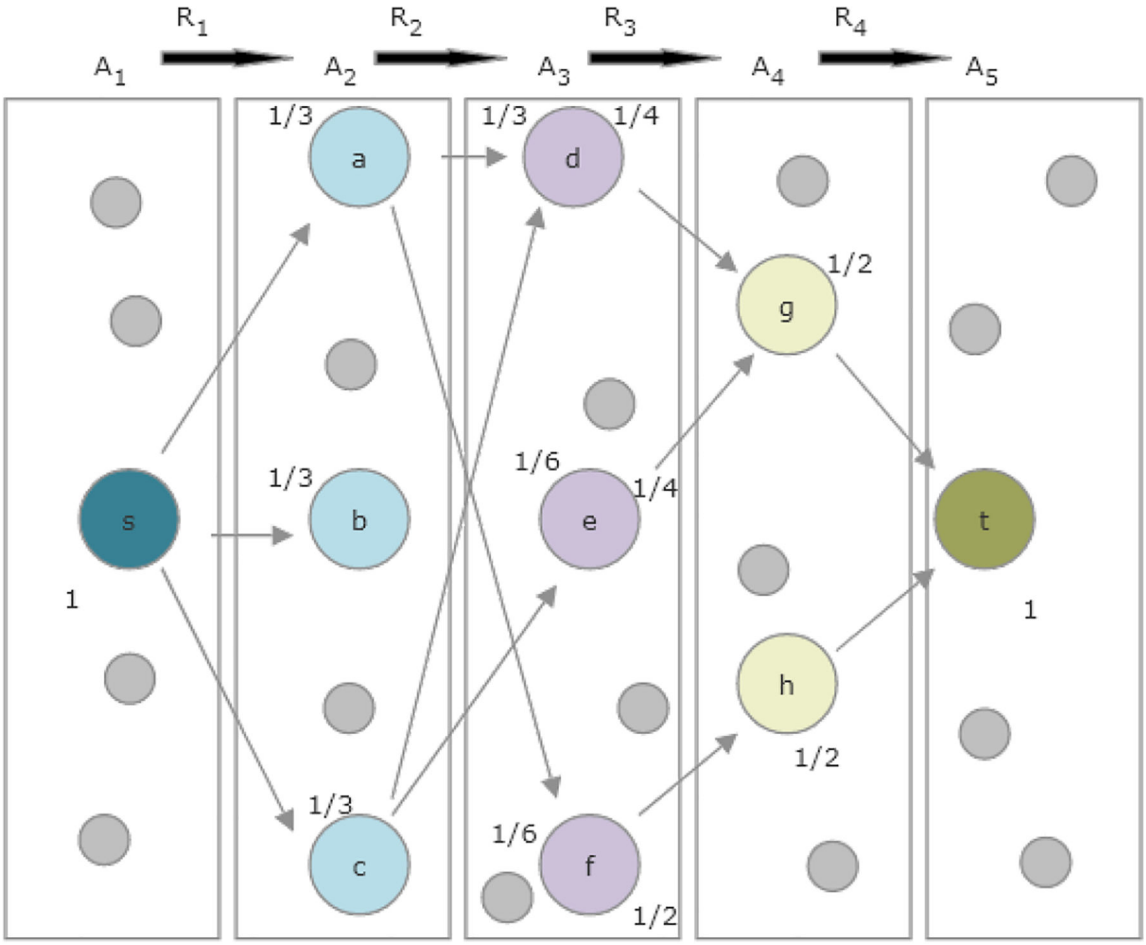

Figure 1.
Example graph, metapath, and HeteSim computation.

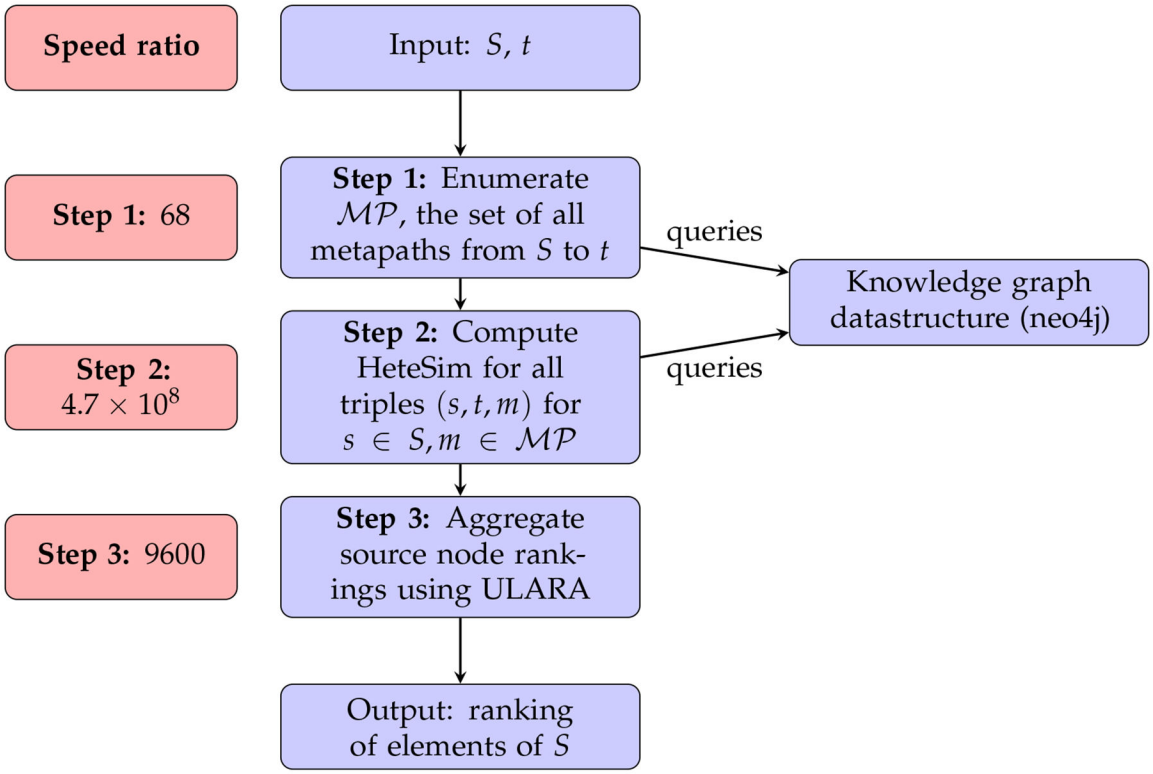

Figure 2.

Overview of SemNet version 1 HeteSim implementation. Speed ratio is computed as (SemNet 1 time)/(SemNet 2 time) and is given for source node insulin and target node Alzheimer's disease. In SemNet 2, the approximate mean HeteSim algorithm is used with approximation parameters $\epsilon = 0.1$ and $r = 0.9$.

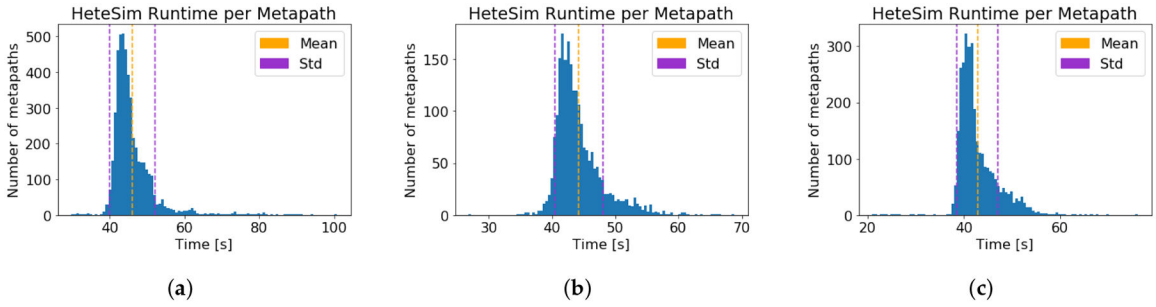

Figure 3.

Distribution of SemNet version 1 HeteSim computation times for all metapaths joining the given source node and Alzheimer's disease. (**a**) Insulin; (**b**) Hypothyroidism; (**c**) Amyloid.

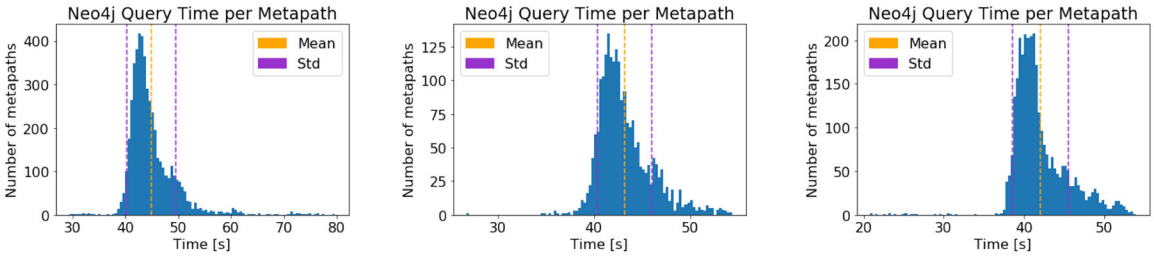

Figure 4.

Distribution of Neo4j query times in SemNet version 1 HeteSim computation for all metapaths joining the given source node and Alzheimer's disease. (**a**) Insulin; (**b**) Hypothyroidism; (**c**) Amyloid.

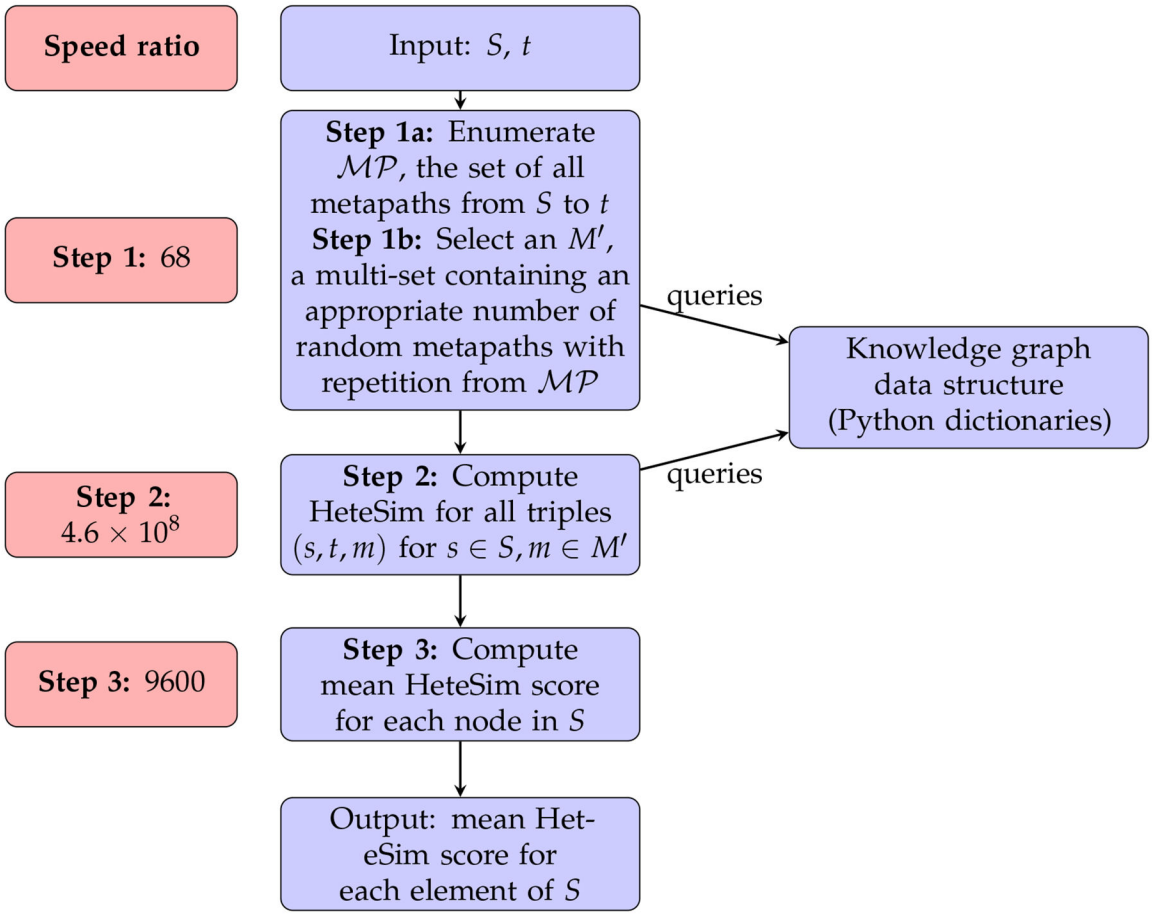

Figure 5.

Overview of SemNet version 2 approximate mean HeteSim implementation. Speed ratio is (SemNet 1 time)/(SemNet 2 time) and is given for source node insulin and target node Alzheimer's disease. SemNet version 2 used approximation parameters $\epsilon = 0.1$ and $r = 0.9$.

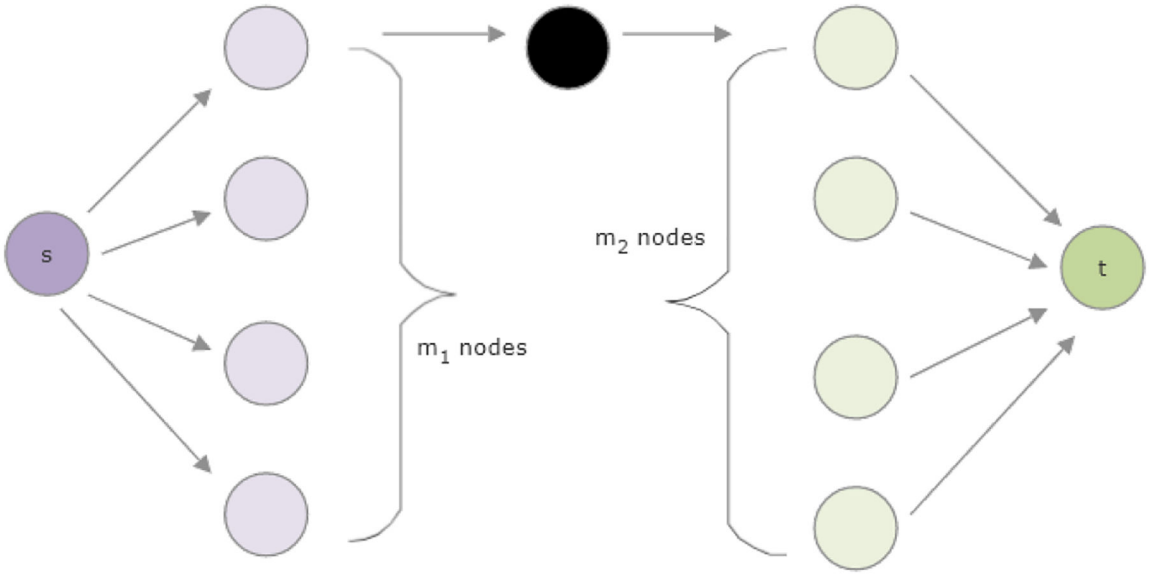

Figure 6.

An example knowledge graph. Here, we use the convention that nodes are organized by type into vertical columns in the order that they appear in the metapath. We also only show edges that may appear in some metapath instance. This example has $m_1 - 1$ dead-end nodes on the left and $m_2 - 1$ dead-end nodes on the right. The HeteSim score of s and t with respect to the metapath is 1 for all values of m_1 and m_2.

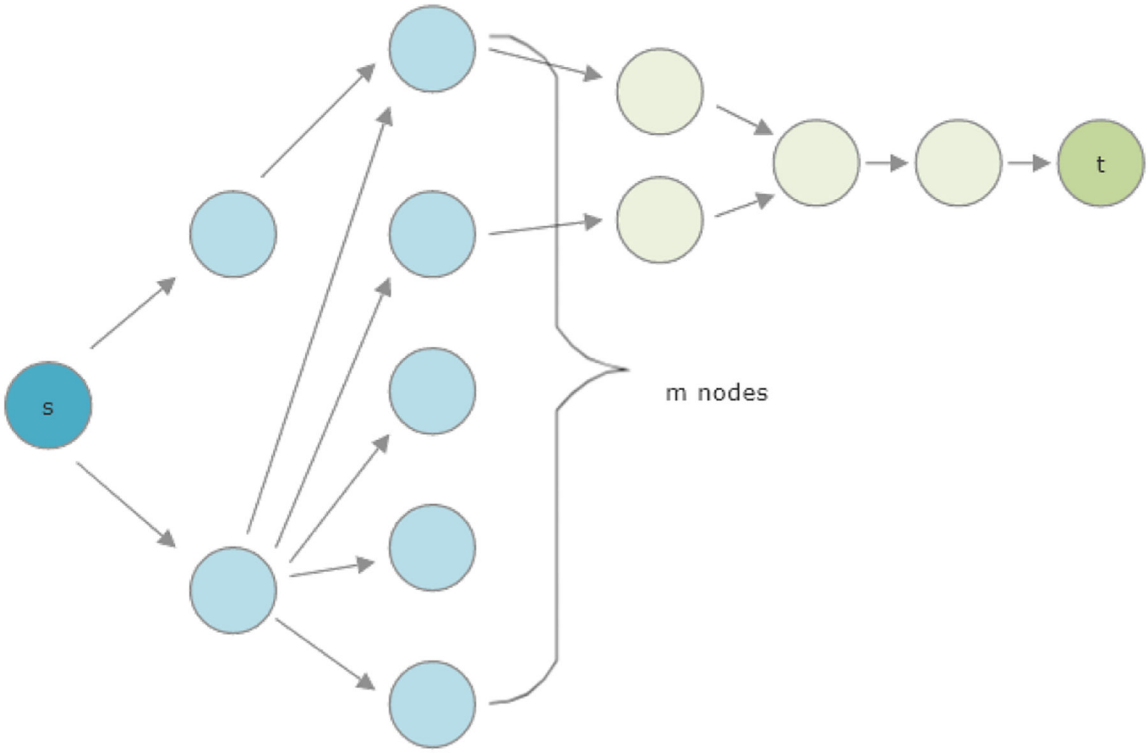

Figure 7.

An example metapath and knowledge graph, drawn with the same conventions as in Figure 6. Note that, in this example, the removal of dead ends does change the HeteSim score.

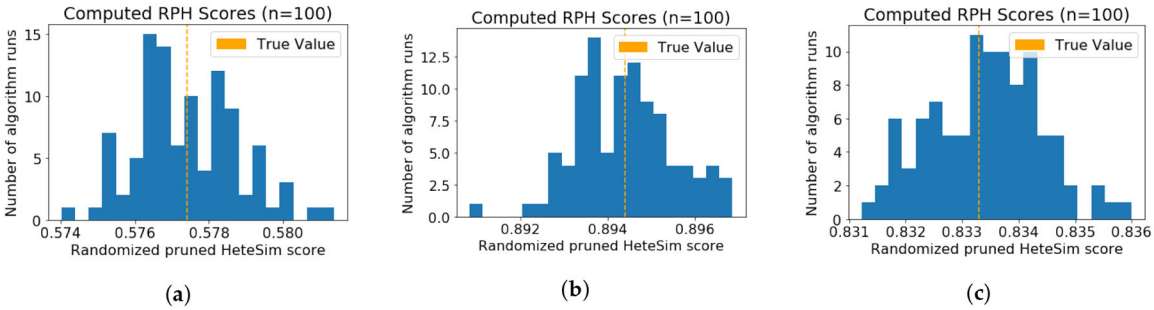

Figure 8.

Computed randomized pruned HeteSim (RPH) scores for each of the three test graphs. (**a**) Test graph 1; (**b**) Test graph 2; (**c**) Test graph 3.

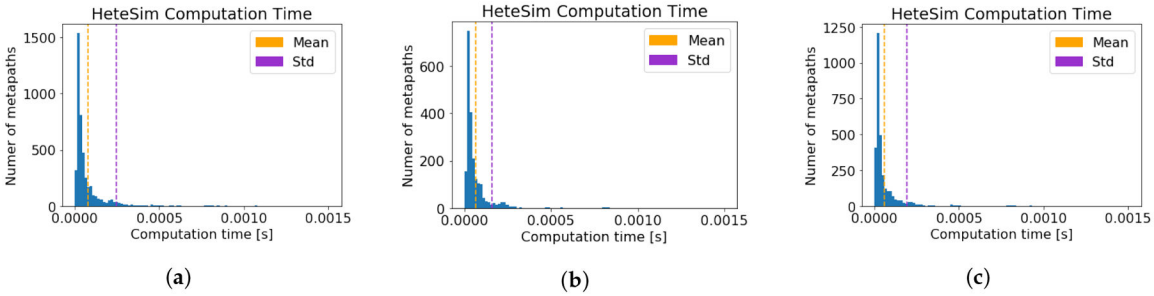

Figure 9.

HeteSim computation times per metapath for all metapaths of length 2 from the given source node to Alzheimer's disease, using the deterministic HeteSim implementation from SemNet version 2. (**a**) Insulin; (**b**) Hypothyroidism; (**c**) Amyloid.

Table 1.

SemNet version 1 HeteSim computation times for all metapaths between each of the three source nodes and Alzheimer's disease.

Source Node	Insulin	Hypothyroidism	Amyloid
Number of metapaths	4873	2148	3095
Total computation time (min)	93.7	39.7	55.4
Computation time per metapath (s) (±std)	46.0 ± 6.1	44.2 ± 3.8	42.8 ± 4.2
Neo4j query time, per metapath (s) (±std)	44.9 ± 4.6	43.2 ± 2.8	42.1 ± 3.5
Time per metapath, excluding query time (s) (±std)	1.1 ± 3.1	1.0 ± 2.3	0.8 ± 1.9

Table 2.

Mean and standard deviation of runtimes for both SemNet version 1 and the approximate mean HeteSim algorithm from SemNet version 2, broken down by step as in Figure 5.

Source Node	Insulin	Hypothyroidism	Amyloid
Num metapaths (SemNet 1)	4873	2148	3095
SemNet 1: Step 1 (s)	81 ± 5.3	35 ± 2.4	84 ± 5.3
SemNet 1: Step 2 (s)	220,000 ± 2300	96,000 ± 270	220,000 ± 2700
SemNet 1: Step 3 (s)	0.80 ± 0.0021	0.39 ± 0.0093	0.80 ± 0.014
Num metapaths (SemNet 2)	4521	2130	3060
SemNet 2: Step 1 (s)	1.2 ± 0.0093	0.19 ± 0.0015	0.41 ± 0.0024
SemNet 2: Step 2 (s)	0.0047 ± 0.00097	0.0026 ± 0.00060	0.0027 ± 0.00061
SemNet 2: Step 3 (s)	$8.3 \times 10^{-5} \pm 1.4 \times 10^{-6}$	$8.3 \times 10^{-5} \pm 1.9 \times 10^{-6}$	$8.3 \times 10^{-5} \pm 1.2 \times 10^{-6}$
Runtime ratio: Step 1	68	184	200
Runtime ratio: Step 2	4.7×10^{8}	3.6×10^{7}	8.1×10^{7}
Runtime ratio: Step 3	9600	470	9600

Table 3.

Mean and standard deviation of runtimes for the mean exact HeteSim and approximate mean HeteSim algorithms.

Algorithm	Runtime (s)
Mean exact HeteSim	4.1 ± 0.060
Approximate mean HeteSim	3.9 ± 0.015

Table 4.

Mean and standard deviation of runtimes for both the deterministic HeteSim and randomized pruned HeteSim algorithms on the top 20 individual length 2 metapaths.

Source Node	Deterministic HeteSim	Randomized Pruned HeteSim
Insulin	$2.0 \times 10^{-3} \pm 1.2 \times 10^{-3}$	3500 ± 3400
Hypothyroidism	$7.2 \times 10^{-4} \pm 3.4 \times 10^{-4}$	440 ± 650
Amyloid	$9.9 \times 10^{-4} \pm 6.4 \times 10^{-4}$	1200 ± 1200

Table 5.

Computation details for the randomized pruned HeteSim algorithm on the top 20 individual length 2 metapaths.

Source Node	Insulin	Hypothyroidism	Amyloid
Max iterations (N)	28,019,926	8,547,987	12,790,378
Min iterations (N)	5,308,942	1,666,564	3,229,242
Mean iterations (N)	10,068,473	2,632,969	5,206,723
Max runtime (s)	14,588	3138	5052
Min runtime (s)	420	99	247
Mean runtime (s)	3491	438	1193
Max metapath instances	488	167	240
Min metapath instances	109	39	70

Table 6.

Maximum, minimum, and mean runtimes (with standard deviation) for the SemNet version 2 deterministic HeteSim algorithm on the top 20 individual length 4 metapaths.

Source Node	Insulin	Hypothyroidism	Amyloid
Max runtime (s)	0.21	0.022	0.033
Min runtime (s)	0.032	0.0029	0.0070
Mean runtime (s) (±std)	0.11 ± 0.039	0.011 ± 0.0056	0.015 ± 0.0075