SemSim: Combining Structural and Semantic Similarity (Full Version)

Tova Milo[‡], Amit Somech[‡] and Brit Youngmann[‡]
[‡]Tel Aviv University
Tel Aviv, Israel
{milo,amitsome,brit}@post.tau.ac.il

Abstract

With the ubiquity of information networks and their wide range of applications, measuring vertex similarity in networks draws extensive interest in various research fields, e.g., social networks and recommender systems. While previous work has mainly focused on two distinct classes of similarity measures, *structural* and *semantic*, we argue that, in practice, a tight integration of the two notions allows to derive a more accurate measure, significantly superior to a simple weighted average of the two. In this paper, we present SemSim, a highly scalable similarity measure interweaving two well studied measures, SimRank, a popular structural measure, and Lin, a common semantic measure. SemSim has an efficient probabilistic framework, anchored in a careful modification of the underlying random surfer-pair model of SimRank. Our model considers all available data, i.e., semantics and edge weights, while preserving the properties necessary for the (refined) application of all state-of-the-art optimizations previously developed for SimRank. Our theoretical and experimental results demonstrate the effectiveness of our measure, compared to previously proposed measures, as well as its efficient and scalable computation.

1 Introduction

Estimating vertex similarity in information networks is a cornerstone of many applications. Prominent examples include social networks that identify users who share similar properties; recommender systems that compute their recommendations based on the similarity between users/items; estimating peer similarity in peer to peer networks and etc. Similarity search is also a fundamental component in many network analysis algorithms in such contexts, such as link prediction, clustering and so on.

Previous work has mainly focused on two separate classes of similarity measures in information networks: *structural* similarity and *semantic* similarity. Structural measures (e.g. [17, 35, 40, 42]), consider solely the structure of the network. More concretely, quantifying structural similarity of two vertices hinges on the evaluation of the compound similarity of their neighbors. Contrarily, semantic and relatedness measures (e.g. [22, 23, 25, 29]), quantify similarity based on similar meaning or semantic content, which play no role in structural-based computations (see Section 6).

The key observation underlying our work is that in numerous practical scenarios, both structural and semantic information is required to properly determine similarity. A naive approach would be to simply consider a weighted average of two measures.

However, we demonstrate that a carefully chosen tighter weaving of the two concepts, leads to superior similarity analysis.

Consider the following illustrative example. In the rectangle in Figure 1 we see a small portion of a bibliographic database (the rest of the figure is irrelevant at this point). The labeled nodes describe authors and terms (fields of interests). There are two types of directed edges - one connects research fields to the authors interested in them, while the other indicates collaborations between authors. Weights on the edges reflect the strength of the relations, e.g., the weight on the edge connecting `Bob` to `Alice` indicates the fraction of their collaborations out of all Bob's collaborations (for the sake of conciseness, some edge labels and weights were omitted). The semantic information (in particular, the concepts' taxonomy) available about the terms, is depicted by the edges and nodes outside the rectangle. We wish to determine which one of Alice and Carol is more similar to Bob.

The data indicates Bob collaborated with both Alice and Carol equally and has in common two fields of interest with each. However, as `Databases` and `Information management` are more common fields of interest than `Crowdsourcing` (we explain how this is quantified in the sequel), shared by `Alice` and `Bob`, it follows from standard argumentation of information theory that an estimation of similarity increases more drastically when indicated by a less frequent event, thus we would expect their similarity score to be greater. Note, however, that considering the *structure of the network* alone (even if we add the taxonomy edges outside the rectangle), Alice and Carol seem equivalent similarity-wise. Indeed, common structural similarity measures [17, 35, 40] would yield the exact same score for both. Contrarily, traditional semantic measures [23, 29], in particular, ontology-based measures, base the similarity assessment relying on the exploitation of taxonomical features, thus, consider only *the nodes connected by the black taxonomy edges*. Thereby, they provide no semantic information for the author nodes. Indeed, all such similarity measures that we examined (see Section 5.1) yield for both pairs identical scores. In contrast, our new measure, SemSim, that tightly weaves together structure and semantics, allows for a refined similarity analysis that distinguishes between the two pairs. (For further details see Example 2.6)

Our work addresses two challenges. The first, as illustrated above, is to design a similarity measure that accommodates both structural and semantic information. The second is performance. Modern information networks are often very large and similarity must be computed efficiently.

SemSim To capture both semantic and structural knowledge, we use a generic simple data model that aligns the information network (e.g. the graph in the rectangle in Figure 1) and the semantic taxonomial knowledge (the graph outside the rectangle) into a single heterogeneous information network (HIN) [22]. Our proposed similarity measure, SemSim, interweaves two well studied measures, Simrank [17], a common structural similarity measure, and Lin [23], a common semantic measure. SimRank is a simple and powerful structural measure, based on a solid theoretical "random surfer" model. It follows a simple and intuitive assumption: *"two objects are similar if they are related to similar objects"*. Lin is a generic *information-content* based semantic similarity measure, that is defined over concept taxonomies and can be computed efficiently. Intuitively, SemSim refines SimRank by weighting, at each step of the computation, the neighbors structural similarity with their semantic similarity scores. (Formal Definition is given in Section 2).

Efficiency Interestingly, not only is SemSim shown to yield superior similarity analysis on multiple real-life examples, it has performance benefits as well. We show through a careful analysis of the mathematical properties of our refined measure that all optimizations previously developed for SimRank and Lin transfer, after appropriate delicate adaptation, to SemSim. Moreover, we demonstrate that the interplay between semantic and structural similarity allows to further speed up the computation. In particular, we show that the semantic similarity of two nodes provides an upper bound on their overall similarity and hence provides natural means to prune "un-promising" vertex-pairs from consideration. As a simple example, consider again Figure 1. Using the semantic similarity bound, it is clear that when searching for highly similar concepts, vertices from semantically unrelated categories are bad candidates (e.g., it is meaningless to compare between `Alice` and `DM`), yet, even concepts that belong to semantically related categories (e.g., `IM` and `CS`) but are far semantically (i.e., their semantic similarity score is relatively low), would yields low SemSim scores, therefore, their intermediate computation may be avoided. We further discuss this in Section 3

We chose SimRank as the basic building block of our work due to its extensive body of available optimizations. Many of these works are based on the connection between SimRank and the "random surfer-pairs" model [12, 18, 31, 37], that is, SimRank can be computed using random walks. A main challenge addressed in our work, was to carefully modify the underlying random walk model to take into account all available data, i.e., semantics and edge weights, while preserving the necessary properties that enable such optimizations. We present an adjusted random surfer model, aware to both semantics and weights, that serves as the foundation for SemSim optimized computation. Building on this model, we provide an optimized, designated to our setting, framework, to efficiently compute approximated SemSim scores and explain how SimRank random-walks style optimization can be employed in SemSim as well. We show that although our model allows to account for more information, thereby yielding a more profound similarity measure, this comes with no significant cost in terms of time and space needed for computing SemSim, compared to the less refined competitors. This is also demonstrated by our experiments.

Interestingly, while some previous work did attempt to consider both semantic and structural knowledge [14, 15, 22, 13], the focus was constrained to specific contexts without addressing the problem in the general case, i.e., quantifying the similarity between vertices in an HIN. Furthermore, the scalability issue was not addressed, that is neither work provided a method for quickly determine the similarity in very large networks. We further discuss this in Section 6.

Our technical contributions (and correspondingly, the paper organization) may be summarized as follows:

1. We use an HIN data model to align structural and semantic information. Based on this simple data model we define SemSim, a refined, generic measure that interweaves structural and semantics (Section 2).
2. We analyze the properties of SemSim showing how the delicate interaction between semantic and structure may be used to speed up computation by pruning irrelevant vertex-pairs (Section 3).
3. We further present an efficient and highly scalable probabilistic framework, extending the analog framework of SimRank, for approximated SemSim computation. The development requires a careful mathematical analysis of SemSim properties and the interaction between its structural and semantic components (Section 4).

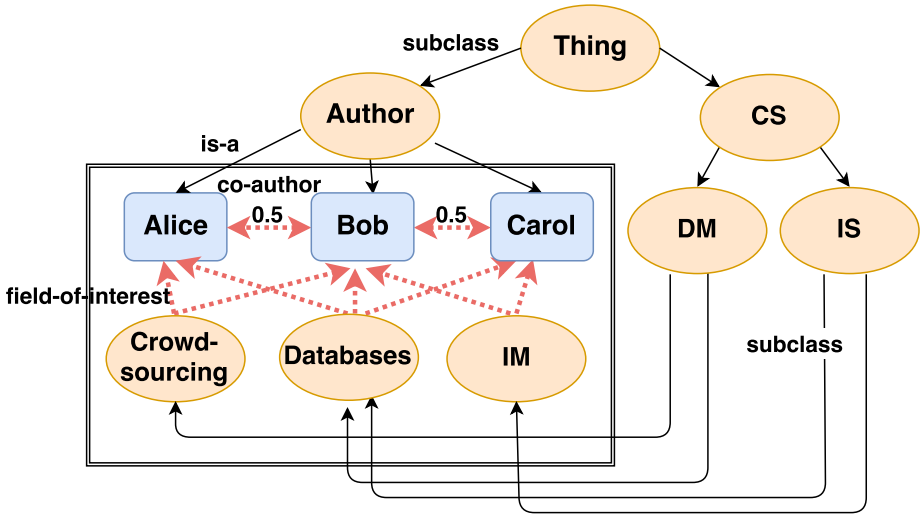


Figure 1: Example HIN, aligning structural relationships (red edges) with semantics taxonomy (black edges).

4. Finally, we conduct an extensive experimental study over real data, demonstrating the effectiveness of SemSim, compared to previously proposed measures, as well as its efficient scalable computation (Section 5).

Related work and conclusions are presented in Sections 6 and 7, resp.

2 Preliminaries

We start with a short background on commonly used structural and semantic similarity measures and their corresponding data models. Then, we explain the combined data model used in our setting and its corresponding novel similarity measure, SemSim.

2.1 Structural Similarity

We start by formally defining the graph model, then provide a short background on SimRank [17].

Following previous works [32] we refer to the objects graph as an *Heterogeneous Information Network*.

Definition 2.1. ***Heterogeneous Information Network.*** *A Heterogeneous Information Network (HIN) is a directed (weighted) graph $G = (V, E, \Phi, \Psi, W)$, where: V is a set of vertices; E is a set of edges; $\Phi : V \rightarrow \mathcal{L}$ is a vertex labeling function; $\Psi : E \rightarrow \mathcal{R}$ is an edge labeling function and $W : E \rightarrow R^+$ is an edge weight function.*

In this paper we consider directed graphs, but all the results can be adapted to undirected graph model with minor modifications. For a vertex v, we denote by $I(v), O(v)$ the set of in and out neighbors of v, respectively. An individual in-neighbor is denoted as $I_i(v)$, for $1 \leq i \leq |I(v)|$, if $I(v) \neq \emptyset$ and $O_i(v)$ the i^{th} out-neighbor, respectively.

SimRank is defined using the following recursive definition, that measures how similar two nodes are by considering the similarity of their in-neighbors.

Definition 2.2. ***(SimRank)*** *Given two vertices a and b in V, their SimRank score is defined as follows:*

$$simrank(a,b) = \begin{cases} 1, & a = b \\ \frac{c}{|I(a)|\cdot|I(b)|} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} simrank(I_i(a), I_j(b)), & a \neq b \end{cases} \quad (1)$$

where c is a decay factor in $[0,1]$. *If* $I(a) = \emptyset$ *or* $I(b) = \emptyset$ *and* $a \neq b$ *then* $simrank(a,b) = 0$.

A solution to Equation (1) can be reached by iterating to a fixed point. For the k^{th} iteration, an iterative similarity function $R_k(a,b)$, denotes the similarity score between a and b on the k^{th} iteration. Initially, $R_0(a,b)$ is defined as 0 if $a \neq b$ and 1 if $a = b$. Iteratively, $R_{k+1}(a,b)$ is computed from $R_k(\cdot,\cdot)$ as follows:

$$R_{k+1}(a,b) = \frac{c}{|I(a)| \cdot |I(b)|} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} R_k(I_i(a), I_j(b))$$

A weighted variant of SimRank, named SimRank++, that takes into account the weights of edges, was presented in [4]. While they focused on a different problem (query rewriting in the context of click graphs), such weighted version is relevant to our context as well. In our setting, W associates to each edge a real (positive) number that represents the strength of the relation (as demonstrated in Figure 1).

2.2 Semantic Similarity

Standard notions of semantic similarity consider the concepts taxonomy (or more generally a partially ordered concept set, represented as nodes of a directed acyclic graph).

A taxonomy is a simple hierarchical arrangement of entities in a graph, which refer to a parent-child kind of relationship as *subclass* or *is-a* relations. The taxonomy edges form a *Rooted DAG* where all nodes are reachable from the root. For example, in Figure 1, the black edges form a taxonomy and the node `Thing` is the root.

Common semantic similarity measures [29, 23] use the notion of *Information content* (IC). Intuitively, the key to the similarity of two concepts is the extent to which they share information in common, indicated by a highly specific concept that subsumes them both. Following the standard argumentation of information theory, the IC of a concept c can be quantified as negative the log likelihood, $-log(p(c))$, where $p(c)$ is the probability of c to occur. Intuitively, as probability increases, informativeness decreases. Previous works [30, 29] suggested tools to build the IC from a given taxonomy that do not require external resources and bounded in $(0,1]$.

Resnik [29] suggested to incorporate IC in the similarity measure as follows: two concepts are more similar if they present a more shared information, and the information shared by two is indicated by the IC of the concepts that subsume them in the taxonomy. A refinement to Resnik's measure, which we adopt in this work, was presented by Lin [23], considering also the IC of each of the evaluated concepts. Intuitively, the similarity between concepts here measures the ratio between the amount of information needed to state their commonality and the information needed to fully describe them.

Definition 2.3. ***(Lin)*** *Given two concepts, a and b ,in a taxonomy, Lin semantic similarity score (denoted as* $Lin(a,b)$*) is defined as follows:* $Lin(a,b) = \frac{2 \cdot Res(a,b)}{IC(a)+IC(b)}$, *where*

$Res(a,b) = -log(\max\{p(LCA(a,b))\}) = \max\{IC(LCA(a,b))\}$ *and* $LCA(a,b)$ *is the set of the lowest common ancestors of* a *and* b

Note that Lin scores are computed using only the taxonomy (that form a rooted DAG) and a given IC function, thus can be computed efficiently (see Section 2.4 for further details).

We conclude with an observation regarding semantic similarity in a taxonomy. We will later use this property when proving the soundness of our new measure (Theorem 2.8).

Observation 2.4. *For every two concepts* a, b, $0 < Lin(a,b) \leq 1$, *assuming the IC of the top concept in the taxonomy* $\neq 0$ *and the IC function is bounded in* $(0,1]$.

2.3 Combining Structure and Semantics

As described in the introduction, our data model merges the (weighted) HIN with semantic taxonomies of the concepts mentioned in the graph.

In the merged graph, the set of edges is composed of two distinct sets: the taxonomy edges and all other edges. To form such a semantic-rich graph, one may use tools enable entity alignment, such as [28], to align an HIN with public available taxonomies, (e.g., the taxonomical parts from [9, 26]), or by using existing partial order over concepts available within the data itself (see Section 5).

Next, we formally define SemSim, a structural-semantic measure, that integrates SimRank and Lin, w.r.t the merged graph. Intuitively, we augment SimRank by both the semantic similarity of nodes as well as the weight on edges between them, to fully account for all available information.

Definition 2.5. ***(SemSim)*** *Given two vertices* a *and* b *in* V, *the* SemSim *similarity score (denoted as* $s(a,b)$*) is defined as follows. If* $a = b$ *then* $s(a,b) = 1$, *else:* $s(a,b) =$

$$\frac{Lin(a,b) \cdot c}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} s(I_i(a), I_j(b)) \cdot W(I_i(a), a) \cdot W(I_j(b), b)$$

where $N_I = \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a), a) \cdot W(I_j(b), b)$, *and* c *is the decay factor* $\in [0,1]$. *If* $I(a = \emptyset)$ *or* $I(b) = \emptyset$ *then* $s(a,b) = 0$.

Following SimRank iterative solution, a solution to Equation (2) can be reached by iterating to a fixed point.

$$R_0(a,b) = \begin{cases} 0, a \neq b \\ 1, a = b \end{cases} \tag{2}$$

$$R_{k+1}(a,b) = \tag{3}$$

$$\frac{Lin(a,b) \cdot c}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} R_k(I_i(a), I_j(b)) \cdot W(I_i(a), a) \cdot W(I_j(b), b)$$

We can now complete the picture for the example in the introduction.

Example 2.6. *Consider again Figure 1, and assume all missing edge weights were set to* 1. *Recall that we wish to determine which one of Alice and Carol is more similar to Bob. If we use only structural similarity, as determined by SimRank, we*

get simrank(Alice,Bob) = *simrank*(Bob,Carol)≈ 0.21. *(For space constraints we omit here the computation details, which can be found in the Appendix. Similarly, if we use only semantics similarity, as determined by Lin, using the relevant IC values (computed in a standard manner as described in Section 5), we obtain Lin*(Alice,Bob) = *Lin*(Bob,Carol)$= 0.9$.

When run on the same data, the computation of Lin scores for the fields-of-interest shared by the authors yields

Lin(Crowdsourcing,DB)$= 0.9$ *and Lin*(DB,IM)$= 0.18$.
Consequently, using SemSim *with the same parameters, we obtain: s*(Alice,Bob)$=$ 0.133*, while s*(Bob,Carol)$= 0.128$ *(the full computation is provided in the full paper) , matching our intuition from the Introduction that Alice's similarity to Bob is greater than Carol's.*

The following observation states the relation between a weighted version of SimRank (as suggested in [4], denoted as SimRank++), SimRank, and our new measure.

Observation 2.7. *If the hierarchical taxonomy is flat, i.e, the semantic similarity function is the constant function* 1*, then* SemSim *is equivalent to SimRank++. If the weight function W is the constant function* 1 *as well, then* SemSim *is equivalent to SimRank.*

To conclude, note that our definition of SemSim considers all neighbor-pairs. An alternative could be to take edge labels into considerations and restrict attention to neighbor-pairs that are pointed by edges having the same label. While such formulation requires only minimal (technical) changes, our experiments showed it to be essentially equivalent, in terms of both accuracy and running times, and we thus omit this restriction.

2.4 Basic properties of SemSim

We next show that SemSim have some desired properties and use them to present a basic algorithm for computing SemSim. This algorithm serves as baseline on which we will improve in the following sections.

Theorem 2.8. *The iterative* SemSim *equations (Equations (3) and (4)) have the following properties:*
1.(Symmetry) $\forall a, b \in V R_k(a,b) = R_k(b,a)$
2.(Maximum self similarity) $\forall a \in V : R_k(a,a) = 1$
3.(Monotonically) $0 \leq R_k(a,b) \leq R_{k+1}(a,b) \leq 1$
4.(Existence) *The solution to the iterative equations always exists and converges to a fixed point, which is the theoretical solution to the recursive equations.*
5.(Uniqueness) *the solution to the iterative equations is unique when* $c \neq 1$.

The proof follows lines similar to that of SimRank in [17], and thus we defer it to the Appendix. We can also show (again, following similar proof for SimRank [43]) that not only the scores are monotone (i.e, $R_k(a,b) \leq R_{k+1}(a,b)$), their differences in consecutive iterations is bounded.

Lemma 2.9. $0 \leq R_{k+1}(a,b) - R_k(a,b) \leq Lin(a,b) \cdot c^{k+1}$

This suggests that the iterative solution for SemSim converges as fast as SimRank, and possibly faster (due to the additional semantic factor). See experimental results.

Another useful property is that the semantic similarity of two vertices provides a natural upper bound on their SemSim score. This property is highly useful, since it can be used to prune un-promising node-pairs.

Observation 2.10. *For every two vertices $a, b \in G$ s.t $a \neq b$: $s(a,b) < Lin(a,b)$.*

Baseline algorithm Note that Theorem 2.8 provides a simple algorithm for computing SemSim, that computes its iterative form to a fixed point (or up to a given precision guarantee). Since the semantic is used as a black-box, we start by recalling from the literature how the semantic similarity is computed, then consider the full computation.

The semantic similarity of two nodes is computed using only the hierarchical taxonomy edges in G (that form a rooted DAG) and a given IC function. To compute the semantic similarity of two concept, one should find their LCA in the taxonomy. Using Tarjan's off-line LCA algorithm for the general DAG case, which pre-processes a taxonomy in linear time, one obtains constant-time LCA queries [16]. Consequently, after $O(|V|)$ pre-processing time, computing $Lin(\cdot,\cdot)$ can be done in constant time.

Given this, we can now analyze the complexity of the iterative procedure for computing SemSim. Let $n = |V|$, c the decay factor and k be the number of iterations executed. The space complexity of the algorithm is $O(n^2)$. Let d be the average in-degree over all nodes of G. Following similar analysis for SimRank, it is easy to show that time complexity of the algorithm is $O(k \cdot d^2 \cdot n^2)$, and the worst case time complexity for a given k is $O(n^4)$. In [24], the authors improved the time complexity of SimRank from to $O(n^3)$, using a memorization technique. The same mechanism can be applied on SemSim to obtain $O(n^3)$ time complexity as well. Finally, another straightforward acceleration is possible: to avoid calculate the same Lin scores more than once, one may maintain a cache mechanism, that stores previous calculations. We will show in Section 5.3 that such caching may accelerate the computation time considerably.

3 Random Surfer-Pairs Model

The baseline algorithm provided in the previous section has two main disadvantages: (i) it computes all pair-wise scores, even if one interested only in a single-pair, and (ii) the complexity is prohibitive for large graphs. To address these issues we provide an alternative interpretation to SemSim, then, explain how SemSim can be computed efficiently.

Jeh and Widom [17] have established a connection to a "random surfer-pairs" model that allows to comput SimRank using random walks. We next show that with appropriate adjustments, an analogous correspondence can be established for SemSim. The key challenge is to incorporate semantic similarity into the random walks. We will show that SemSim measures how soon two random surfers are expected to meet, if they start in two nodes and randomly walk the graph backwards, while being aware to both edge weights and semantics. We start by defining semantic-aware random walks, then explain how SemSim can be computed using such walks.

3.1 Semantic-Aware Random Walk

To define semantic-aware random walk formally, following [17], we use the definition of a *node-pair graph* G^2, in which each node represents an ordered pair of nodes. An edge $((a,b),(c,d)) \in G^2$ iff, $(a,c),(b,d) \in G$. We extend the definition with an assignment of weights: the weight of an edge $e = ((a,b),(c,d))$ is: $W(e) = W(a,c) \cdot W(b,d)$.

Let us assume that all edges in G have been reversed, so following an edge is equivalent to moving one step backwards in the original graph. For example, Figures

2a and 2b display a sample graph G and all out-edges from node the `(A,B)` (after reversed). We call a node $(u, v) \in V^2$ a *singleton node* if it represent the same node from V, e.g., the node `(Author,Author)` in Figure 2b.

In SimRank, the random surfer chooses the next neighbor uniformly, out of all out-neighbors of the current state. In our case, the surfer must account both weights and semantics. Thus, the probability a surfer (traveling G^2) in current node (a, b) would move next to it out-neighbor (c, d) is defined as: $P[(a,b) \to (c,d)] := \frac{W((a,b),(c,d))}{\sum_{k=1}^{|O((a,b))|} W((a,b),O_k(a,b))}$

To account the semantics as well, we multiply the probability with the semantic score of the current node, i.e., $Lin(a, b)$. A path in G^2 represents a pair of paths in G. Let $t = \langle w_1, ..w_k \rangle$ be a path in G^2, where $l(t) = |t|$, in this case $l(t) = k - 1$, has the probability $P[t]$ of traveling within it, where $P[t] := \prod_{i=1}^{k-1} P[w_i \to w_{i+1}]$, or 1 if $l(t) = 0$. To derive the path's weight, we multiply each step with its corresponding Lin score, that is, $W[t] := \prod_{i=1}^{k-1} P[w_i \to w_{i+1}] \cdot Lin(w_i)$

We next explain the connection between semantic-aware walks over G^2 and SemSim scores, based on the relation between random-walks and SimRank. Intuitively, we would prove that the SemSim score of a node $(a, b) \in V^2$, can be computed using all paths from (a, b) leading to a singleton node in G^2. Let $t : (a, b) \rightsquigarrow (x, x)$ be the set of paths in G^2 form (a, b) to all singleton nodes, namely, t iterates over all paths from (a, b) to some node (x, x). By definition, (x, x) is the first and only singleton node in t (after the first meeting, the two surfers halt). Let: $s'(a,b) = \sum_{t:(a,b)\rightsquigarrow(x,x)} W[t] \cdot c^{l(t)}$ Theorem 3.1 provides an alternative way to compute the SemSim score for a given pair of nodes, using semantic-aware walks over G^2.

Theorem 3.1. $s'(a, b) = s(a, b)$

Proof. If $a = b$ then $s'(a, b) = s(a, b) = 1$. Else, if there is no path in G^2 from (a, b) to any singleton node, then $s'(a, b) = 0$, and it is easy to see that $s(a, b) = 0$ as well. Otherwise, consider a path t from (a, b) to some singleton node, in which the first step is the out-neighbor $O_i((a, b))$. Denote t' the path from $O_i((a, b))$ to the singleton node. Let T be the bijection that takes each t' to the corresponding t, by appending an edge at the beginning, i.e., if $l(t') = k$, then $l(t) = l(T(t')) = k + 1$.

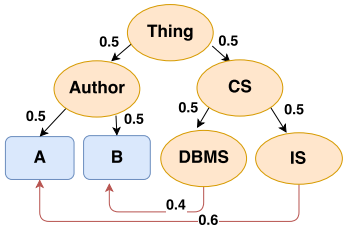
The weight of the path t is $W[t] = P[(a,b) \to O_i((a,b))] \cdot Lin((a,b), O_i((a,b))) \cdot W[t']$. We can now split the sum in $s'(a, b)$ according to the first step of t to write:

$$s'(a,b) = \sum_{j=1}^{|O((a,b))|} \sum_{t':O_i((a,b))\rightsquigarrow(x,x)} W[T(t')] \cdot c^{l(T(t'))} =$$

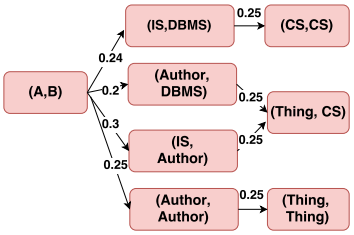
$$\sum_{j=1}^{|O((a,b))|} \sum_{t':O_i((a,b))\rightsquigarrow(x,x)} P[(a,b) \to O_j((a,b))] \cdot Lin((a,b), O_i((a,b))) \cdot W[t'] \cdot c^{l(t')+1} =$$

$$\frac{Lin(a,b) \cdot c}{\sum_{i=1}^{|O(a,b)|} W((a,b), O_i(a,b))} \sum_{i=1}^{|O((a,b))|} \sum_{t':O_i((a,b))\rightsquigarrow(x,x)} W[t'] \cdot W((a,b), O_i(a,b)) \cdot c^{l(t')} =$$

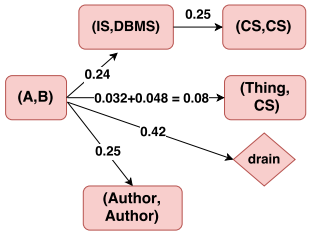
$$\frac{Lin(a,b) \cdot c}{\sum_{i}^{|O(a)|} \sum_{j}^{|O(b)|} W(a, O_i(a)) \cdot W(b, O_j(b))} \sum_{i=1}^{|O(a)|} \sum_{j=1}^{|O(b)|} s'(O_i(a), O_j(b))$$



(a) Sample graph G



(b) All paths from `(A,B)` in G^2.



(c) All paths from `(A,B)` in G^2_θ.

Figure 2: Example of the reversed pair-node graphs G^2 and G^2_θ, according to G.

The last equation is identical to SemSim equation (Equation 2), where in-edges swapped for out-edges. Since the solution to SemSim is unique (as proved in Theorem 2.8), $s'(a,b) = s(a,b)\ \forall a,b \in V$. □

Using our refined model, one may easily compute the similarity scores on G^2. However, for large graphs the size of G^2 may be too large. We next explain how semantic information may be employed to overcome this problem. First, we will show that it can be effectively used to reduce the size of the considered graph. Second, we will show that the semantic-aware walks may in fact be also performed directly on G. The latter result follows an analogous observation for SimRank [19, 12], but a careful refinement is required to incorporate weights and semantics.

3.2 Reducing the size of G^2

In many practical applications one is interested only by node-pairs whose similarity scores are above a certain minimal threshold. Semantic similarity provides an efficient tool to prune G^2 in such situations. Intuitively, Observation 2.10, provides a semantic-based upper bound on the similarity scores, which can be used to avoid materializing un-promising vertex-pairs. We next define a reduced version of G^2 on which the computation of SemSim scores (for node-pairs with similarity higher than a given threshold) yields the same result as that computed via the full graph G^2.

Given a threshold θ, we next define G^2_θ, a vertex-pair graph which includes only pairs s.t their *Lin* score are $> \theta$. Intuitively, each path from G^2 that is not included in G^2_θ is replaced by a corresponding edge, whose weight reflects the weight of the omitted path. If such an edge already appeared in G^2, the weight is added to the original edge weight. Moreover, the weight of omitted paths form G^2 are weigh by the decay factor power their length, to ensure the similarity scores would not be effected by paths that got shorter in G^2_θ. The graph G^2_θ includes a new vertex d, that has only in-neighbors and

used as a “drain", i.e., it ensures that the probability of choosing a neighbor remains the same.

Definition 3.2. *(G^2_θ.) Given a node-pairs graph G^2 and a threshold $0 < \theta < 1$, $G^2_\theta = (V_\theta \cup \{d\}, E_\theta, W_\theta)$, where: $V_\theta \subseteq V^2$ is a set of vertices and d is a new vertex, E_θ is the edges set and W_θ is a weight function, defined as follows.*

- *A node $(a, b) \in V_\theta$ iff $Lin(a, b) > \theta$.*
- *An edge $e = ((a, b), (c, d)) \in E_\theta$ iff at least one of the following conditions holds. (1)They are adjacent in G^2 (2)There is a path $\langle (a, b), w_1, ., w_k, (c, d)\rangle \in G^2$ and $w_1, ., w_k \notin V_\theta$.*
- *The weight of an edge $e = (u, v)$ defined as $W_\theta(e) = W_1(e) + W_2(e)$ where:*
  $W_1(e) = W(e)$ if $e \in G^2$, $u, v \in V_\theta$ and 0 otherwise.
  $W_2(e) = \sum_{t:u \rightsquigarrow v, t \in G^2} P[t] \cdot c^{l(t)-1}$ where $t = \langle u, w_1, ..w_k, v\rangle$ and $w_1, ..w_k \notin V_\theta$
- *For the new vertex d, edges and weights are defined as follows. $\forall u \in V_\theta$, $(u, d) \in E_\theta$ if $\sum_{j=1}^{|O(u)|} W(u, O_j(u)) \neq \sum_{j=1}^{|O(u)|} W_\theta(u, O_j(u))$ and $W_\theta(u, d)$ is the difference between the sums.*

In the last inequality, we can prove that the left hand side is always grater or equal than the right, therefore the weights are always positive.

Additional pruning of edges can be done by the removal of all out-edges from singleton nodes. Since only paths that reaches exactly once to a singleton node may effect the similarity score, such edges can be omitted (e.g., the edge between `(Author,Author)`) and `(Thing,Thing)` has been omitted in Figure 2c). The following example illustrates a part from the graph G^2 and its corresponding graph G^2_θ.

Example 3.3. *Consider Figures 2b and 2c. Assume $c = 0.8$, Lin* `(Author,DBMS)` *= Lin* `(Author, IS)` *$= 0.2 < \theta$, and Lin scores of all other nodes > theta . Next, the in-neighbor* `(A,B)` *of* `(Author,DBMS)` *is connected to it out-neighbor,* `(Thing,CS)`*, with the weight: $\frac{0.2 \cdot 0.2}{0.99} \cdot \frac{0.25}{0.25} \cdot 0.8 \approx 0.032$. Similarly,* `(A,B)`*, is connected again to* `(Thing,CS)` *with additional weight of 0.048, therefore, the total sum to 0.08. Since the sum of all out-edges weights from* `(A,B)` *has changed, additional edge is added to the drain, with weight that reflects this difference (in this case $0.99 - 0.57 = 0.42$).*

The similarity score over G^2_θ, denoted as $s_\theta(\cdot, \cdot)$, is defined as the result of the random surfing computation on the reduced graph. Namely, if $(a, b) \notin V_\theta$ then $s_\theta(a, b) = 0$, else $s_\theta(a, b) = \sum_{t:(a,b) \rightsquigarrow (x,x)} W[t] \cdot c^{l(t)}$, where t is a path in G^2_θ. Theorem 3.4 provides an alternative way to compute SemSim scores, using semantic-aware random walks over G^2_θ. For lack of space, we only provide proof sketch.

Theorem 3.4. $\forall a, b \in V_\theta : s_\theta(a, b) = s(a, b)$

Proof. (sketch.) First note that all singleton nodes are in G^2_θ. Furthermore, every path $t : (a, b) \rightsquigarrow (x, x)$ in G^2, t is either a path in G^2_θ as well, or t is not a path in G^2_θ, that is, at least one of its nodes not in V_θ. By definition of G^2_θ, each weight of omitted edge from t, is added to a new edge in G^2_θ, or added to the weight of an existing one. That is, each path from G^2 that is not in G^2_θ, is either merged with existing path (that its weight is the sum of all merged paths) or got shorter in G^2_θ (i.e, part of its edges where omitted), yet, its weight remained the same. Formally, one could map all paths $t : (a, b) \rightsquigarrow (x, x)$ in G^2 to their corresponding (perhaps smaller) set of paths in G^2_θ, where if two or more paths are mapped to a single path in G^2_θ, its weight reflects the sum of all merged paths.

The weight of a path from a node to any singleton node in G^2_θ, is consist with the one in G^2, because of the new vertex d. Importantly, by the definition of G^2_θ, each edge

represent a path from G^2, reflects the entire path weight, multiply by the decay factor power its length. Therefore, since each path that contribute to the similarity score of (a, b) in G^2 is represented in G^2_θ as well, computing SemSim, using semantic-aware walks over G^2, is equivalent to the computation over G^2_θ, for all nodes $(a, b) \in V_\theta$. □

To avoid materializing G^2 when obstructing G^2_θ, it can be directly constructed from G. For space constraints we only sketch the procedure. Intuitively, this is done by replacing every path consisting entirely of unwanted pair-vertices, except from the first and last vertices, by an edge. Then, adding edges from every vertex to the drain if needed, while not adding out-edges from singleton nodes. To answer quickly if such path exists, one can use as a pre-processing phase graph reachabilities methods such as [8], or alternatively, iterative deepening depth-first search, in case not many pair-nodes admitting the threshold.

3.3 Computing SemSim directly on G

The reduced graph G^2_θ has two key advantages (i) its size is smaller than that of G^2 and may be tuned by choosing the threshold θ and (2) computing SemSim over G^2_θ requires exploring fewer and shorter paths, hence is more efficient. However, in some cases, especially when considering very large graphs, even our compact representation might still be excessively large. In such cases, an alternative approach that simulates two random surfers directly over G may be used. This approach was originally proposed for SimRank, but some careful adjustments are required in our settings.

Given a node $w_1 \in V$, a reverse random walk from w_1 is a sequence of nodes $t = \langle w_1, .., w_k \rangle$, s.t w_{i+1} is selected uniformly at random from the in-neighbors of w_i. Suppose that we have two reverse random walks t_1 and t_2 from two nodes u and v, respectively, and they first meet at the τ-th step, i.e., the τ-th steps of t_1 and t_2 are identical, but for any $l < \tau$, the l-th steps are different (if the two walks do not meet, $\tau \to \infty$). Jeh and Widom [17] established the following connection: $simrank(u, v) = E[c^\tau]$. As we later explain, this is the key idea for many SimRank optimizations.

Adjustments required for SemSim In SemSim case, rather than uniformly choosing the next step, the random surfer must be aware of both the semantics and weights. Therefore, we cannot just use the first meeting point of two random walks - the weight of each semantic-aware walk need to be consider as well. That is, given two reverse random walks t_1 and t_2 that start from two nodes u and v, respectively, and they first meet at the τ-th step, we can compute the weight of this semantic-aware walk as described in Section 3.1. Thus, given two random walks $t_1 = \langle u_1, .., u_k \rangle$ and $t_2 = \langle v_1, .., v_k \rangle$ of length $k - 1$ we denote t, the coupled random walk of t_1 and t_2, where $t = \langle (u_1, v_1), .., (u_k, v_k) \rangle$ (following [12]). Denote $\tau(t)$ the prefix of t until the first meeting point (inclusive), or t itself in case there is no such point. In addition, denote: $s^*(u, v) = \sum_{t:(u,v)\rightsquigarrow(x,y)} W[\tau(t)] \cdot c^{l(\tau(t))}$, where t is a coupled random walk and $l(\tau(t))$ is the length of the prefix $\tau(t)$. Since only walks who indeed meet increases the sum, we get: $s^*(u, v) = \sum_{t:(u,v)\rightsquigarrow(x,y)} W[\tau(t)] \cdot c^{l(\tau(t))} = \sum_{t:(u,v)\rightsquigarrow(x,x)} W[\tau(t)] \cdot c^{l(\tau(t))}$.

The main difference when considering the coupled walks directly on G, as appose to G^2, is that the prefixes must be distinct, i.e., even if two coupled walks are different, yet their prefixes until the first meeting point are identical, we want to consider this prefix exactly once. Thereby, $s^*(u, v) \notin [0, 1]$, because if we consider the same prefix more than once, we may end with a score larger than 1, therefore, an adjustment is required. Formally, denote: $T = \{\tau(t) : (u, v) \rightsquigarrow (x, y)\}$ i.e., T is the set of all prefixes

of coupled walks that first meet in some singleton node. According to Theorem 3.1 we get: $\sum_{t \in T, t:(u,v) \rightsquigarrow (x,x)} W[t] \cdot c^{l(t)} = s(u,v)$

4 Approximated SemSim

In the previous section we discussed an alternative interpretation of SemSim, based on semantic-aware walks. This technique provides in particular means to compute single-pair SemSim queries, without computing all pair-wise scores. However, for large graphs it may still be expensive, since one considers *all* paths to *all* singleton nodes, even if such paths are long or numerous. In this section, we revisit a major approach to efficiently approximate SimRank, using random walks, then explain the critical adjustments required to make this work for SemSim.

The Monte-Carlo approximation framework utilizes the concept of reverse random walks and the fact that SimRank score can simply be approximated by using the average length of the samples walks [12]. Many of SimRank previous works suggested optimizations based on this technique [37, 18, 31]. This framework precomputes a set W_i of reverse random walks from each node v_i in G, such that (i) each set has the same number, n_w, of walks, and (ii) each walk in W_i is truncated at step t, i.e., the nodes after the t-th step are omitted. Then, given two nodes v_i and v_j, the method estimates their SimRank score as $\hat{simrank}(v_i, v_j) = \frac{1}{n_w} \sum_{l=1}^{n_w} c^{\tau_l}$ where τ_l denotes the step at which the two walks first meet, and ∞ otherwise.

Applying the Monte-Carlo Method to SemSim For SemSim, a different approach is required, since the expected value of c^τ is not equal to SemSim, contrary to SimRank. The idea underlying our solution is that instead of taking the average obtained score of all coupled random walks, we approximate SemSim by summing up all sampled walks. As a result, the error of our approximation depends only on paths we did not sample. Intuitively, as in SimRank, we first sample random walks from each node. The main difference is the way we calculate similarity scores using the obtained random walks. Given a query, we sum the weight of all distinct prefixes of coupled walks, and this sum would serve as the query output.

Algorithm 1 provides a framework for Monte-Carlo simulation to compute single-pair SemSim scores. At pre-processing phase, we generate n_w random walks from each node (lines $1-6$). Then, when a single-pair query is given, we start by collecting all distinct prefix coupled random walks that indeed meet (lines $3-10$). Eventually, for each such prefix, its weight is computed (as defined in Section 3.1) and multiplied by the decay factor power it length (lines $11-14$). We next analyze the complexity of this algorithm and then establish an upper bound on the expected error.

Complexity The offline phase of the framework requires generating n_w random walks from each node, and pre-processing the taxonomy for LCA queries (as explained in Section 2.4), therefore, requires $O(n \cdot n_w \cdot t)$ time and space. Answering a single-pair query requires $O((n_w)^3 \cdot t)$ time, since one must compute the set of all distinct prefixes (lines $3-9$), which requires $O((n_w)^3 \cdot t)$ time (for each prefix we must check if it is already considered, using amortized analysis for lines $8-9$), finally, summing all prefixes weights (lines $10-11$).

One may reduce the space complexity by adopting techniques previously suggested for SimRank (for efficient storage of walks). For example, in [31] the authors suggested to use *one-way graphs*, a compact data structure, as index.

Error The error depends on the following parameters: (1) the decay factor; (2) the number of sampled walks per node; (3) the length of each walk; (4) the maximum out-degree and (5) the maximal ratio between two edges weights, outgoing from the same source. For simplicity, we next provide a theoretical analysis of the expected error in the worse case. In practice, as we demonstrate in Section 5.3, the error is significantly smaller.

Denote d the maximum out-degree in G. The maximum number of paths of length t, from a single node, is bounded by d^t. Therefore, the probability of a single random walk to be selected is $\frac{1}{d^t}$, assuming the paths are selected uniformly. Thus, the probability of a single coupled random walk, coming from a node in G^2, to be considered is $\frac{1}{d^{2t}}$ Note that the error increases for every miscounted coupled walk, if indeed, it reached a singleton node in G^2, i.e., the two random walks met. Furthermore, the two random-walks could meet at any index $i \in [1, t]$. We next analyze the error of approximation according to the meeting index.

First, note that every coupled walk that first met after the t^{th} step would not be considered, therefore, the total error of all such path is bounded by c^{t+1}, because the total weight of all such paths is bounded by 1. Next, we analyze the error caused by a coupled walk of length t, in which its first meeting point is at index i for $1 \leq i \leq t$. Denote k the maximum ratio between two edges weight coming from the same source. For simplicity, assume that the out degree of every node is exactly d, and from every node there is at least one out-edge with weight k and at least one edge with weight 1. Those assumptions assume the worse case, therefore would provide an upper bound on the approximation error. Hence, given a path with a first meeting index i, its maximal weight is bounded by $(\frac{k}{k+d-1})^i$, therefore miscounting this path would lead to a maximal error of: $(\frac{c \cdot k}{k+d-1})^i$. The probability that such path was not considered is $(1 - \frac{1}{d^{2i}})^{n_w}$, since we generate n_w random walks from each node and consider all pair-wise coupled walks. Therefore, the expected error of paths that first meet at index i is: $(1 - \frac{1}{d^{2i}})^{n_w} \cdot (\frac{c \cdot k}{k+d-1})^i$. Consequently, the expected error is bounded by:

$$\sum_{i=1}^{t}((1 - \frac{1}{d^{2i}})^{n_w} \cdot (\frac{c \cdot k}{k + d - 1})^i) + c^{t+1} \tag{4}$$

Using the Cauchy-Schwarz Inequality[1], we get that Expression (4) is bounded by:

$$\sum_{i=1}^{t}((1 - \frac{1}{d^{2i}})^{n_w} \cdot \sum_{i=1}^{t}(\frac{c \cdot k}{k + d - 1})^i) + c^{t+1} \tag{5}$$

Let $\delta = \frac{c \cdot k}{k+d-1}$. Note that the left hand sum get its maximum when $i = t$ and the second is a geometric progression, hence, Expression (5) is bounded by:

$$t \cdot ((1 - \frac{1}{d^{2i}})^{n_w} \cdot \frac{\delta \cdot (\delta^t - 1)}{\delta - 1} + c^{t+1} \leq t \cdot e^{\frac{-n_w}{d^{2t}}} \cdot \frac{\delta \cdot (\delta^t - 1)}{\delta - 1} + c^{t+1}$$

Note, however, that this bound is not tight for the following reasons: (1) the semantic similarity is not considered. Each path weight consider also the semantic similarity of its nodes (for simplicity, we bounded the semantics by 1) and (2) in practice, not every path leads to a singleton node, therefore miscounting it does not increases the error (recall that we assumed every miscounted path would increase the error). In Section 5.3, we report the approximate theoretical error and the maximal error in practice, on real datasets.

[1] We used a simpler and less tight version of the inequality, omitting the square root and power by 2 (all numbers are necessarily positive), for simplicity.

Input : n_w=number of walks, t=path length, c=decay factor, G the input graph.

RW = { }
foreach $u_i \in G$ **do**
  W_i = { }
  for j = *1,..*n_w **do**
    w = reversed random walk of length t starting from u_i, $W_i = W_i \cup \{w\}$
  $RW = RW \cup W_i$
Query `sim`(u, v)
  $sim = 0$, $P_{u,v}$ = { }
  for i,j = *1,..*n_w **do**
    Let $w_{i,j}$ the coupled walk of the i-th walk from u and the j-th walk from v
    Let k be the samllest offset s.t the i-th walk from u and the j-th wlak from v meet
    if *such k exists* **then**
      Let $\tau(w_{i,j})$ the prefix of $w_{i,j}$ up to offset k
      if $\tau(w_{i,j})$ *not in* $P_{u,v}$ **then**
        $P_{u,v} = P_{u,v} \cup \{\tau(w_{i,j})\}$
  foreach p *in* $P_{u,v}$ **do**
    Denote $p = \langle (u_1, v_1), ..(u_k, v_k) \rangle$, $s = 1$.
    for $i = 1, ..k-1$ **do**
      $s = s \cdot \frac{Lin(u_i,v_i) \cdot W(u_{i+1},u_i) \cdot W(v_{i+1},v_i)}{\sum_{j=1}^{|I(u_i)|} \sum_{z=1}^{|I(v_i)|} W(I_j(u_i),u_i) \cdot W(I_z(v_i),v_i)}$
    $sim = sim + s * c^{l(p)}$
  return *sim*

Algorithm 1: Monte-Carlo based framework for SemSim.

Remarks We conclude with two remarks. Up to this point, we assumed that the random walks were constructed uniformly, i.e., the next step was chosen uniformly out of all in-neighbors of the current state. In some cases, generating random walks corresponding to the weight function may be preferable, since in this case the probability of not sampling a "heavy" path is lower. In Section 5.3, we test and report the error using the two strategies.

Finally, a large portion of SimRank's approximation-based optimizations can be refined for SemSim as well, using the adjusted random model and our new evaluation of approximated similarity scores. For example, in [37], the authors provided new interpretation of SimRank, based on $\sqrt{c}$-*random-walks*, which ensures that the expected length of walks is small. This interpretation can be applied to our setting as well (using semantic-aware $\sqrt{c}$-random walks).

5 Experimental Results

We conducted all of experiments on a Linux server with a 2.1GHz CPU and 94GB memory. All methods tested are implemented in Java 7 using JGraphT (`http://jgrapht.org/`), a Java package for analysis of large networks. This package includes an implementation of Tarjan's LCA algorithm which we used in our implementation. As described in Section 2.4, we implemented a simple cache mechanism, consisting of the last $|V|$ LCAs results.

Evaluation Aspects To quantitatively evaluate the proposed formulations, we examine two main aspects. (1) We demonstrate the advantage of our measure, compared

Table 1: Datasets.

Dataset	Small version	Large version
Wikipedia	n = 4.7K m =101K	–
AMiner	n = 7.8K m =47K	n = 0.35M m =3M
Amazon	n = 14.5K m =62K	n = 0.6M m =6M
WorNet	n = 82K m =128K	–

to previous structure-only or semantic-only measures, for capturing objects similarity in practical settings. (2) We examine our sampling-based optimized algorithm performance in terms of execution time and accuracy of approximation, and demonstrate the effectiveness of our pruning method.

Data-sets In our experiments we used several graph datasets, commonly used in the literature, which are suitable for our settings, i.e., have natural related similarity questions, and include objects that posses both structural information and semantic meaning. Table 1 shows the size of each graph. In all datasets, the IC was computed using the Seco formula [30]. In datasets where edge weights were not available, we set the weights following uniform distribution among all outgoing edges of the same type, from each node. We next briefly describe each dataset. **AMiner.** This graph is extracted from [3], which contains data about 1.5M academic papers. From the original citation data, we extracted a weighted co-author graph focused on 30 databases conferences (e.g, ICDE). From each paper, we extracted its authors and relevant terms. The domain taxonomy was built by aligning the terms to concepts from DBpedia [9], and constructing the corresponding taxonomy (as in Figure refex1). The graph includes edges of three types: (1) collaboration edges (with weights reflecting the fraction of one's publications with a given co-author among all of her publications); (2) edges between terms and authors (where an edge's weight correspond to the prevalence of the term in a given author's papers) and (3) taxonomy edges. **Amazon.** This dataset was obtained from [20] and contains 0.5M items of different categories and information about their co-purchases in a given month, for several months. The domain taxonomy was built by Amazon categories data as available in the dataset. The graph contains two types of edges: (1) edges between co-purchase items (with a weights reflecting the fraction of times two item were bought together among all of the times the item was bought) and (2) taxonomy edges. **Wikipedia.** This dataset, also from [20], contains 4.7K Wikipedia articles, represented by nodes in the graph. The domain taxonomy was built by Wikipedia categories as available in the dataset. The graph contains two types of edges: (1) links between articles and (2) taxonomy edges. **WorNet.** The dataset is the noun sub-part of the lexical base WordNet [26]. The edges types are:(1) part-of relations, the non-hierarchical relations available in WordNet (e.g, *meronym-part*, etc.) and (2) taxonomy edges.

For both the AMiner and Amazon datasets, we built small and large versions (see Table 1). In AMiner, the small version includes the top 7K authors with the largest number of publications, and in Amazon, it includes the top 5K most bought items. The smaller versions were used for computing exact similarity scores, while the larger versions were used in our scalability experiments where the approximated computation was tested.

5.1 Experimental Setup

We shortly describe the competing measures that we examined and the parameters setup.

Competing similarity measures To highlight the advantages of combining semantic and structural information in similarity estimation, we compared SemSim with several commonly used structural and semantic similarity measures, in terms of results accuracy.
(1) SimRank [17] as describe in Section 2.1. **(2) SimRank++ [4]** In their work, the authors considered bipartite graphs that model sponsored search, and takes into account the *evidence* supporting query similarity in addition to the edges weights. Since no queries are available in our setting, and we do not assume a bipartite graph, we consider only the weights, which in this case amount to a restricted version of SemSim ignoring semantics. **(3) Lin [23]** as described in Section 2.2. In particular, this measure ignores the structural relation edges. **(4) Average** an average of the obtained Lin and SimRank scores (we omitted results for average scores of other semantic and structural measures, as our conclusions for these measures also favor SemSim based on qualitatively identical arguments). **(5) PathSim [35]**, an alternative structural similarity measure, based on random-walks. It is designated for HIN, and considers the edge labels as well. **(6) Panther [40]**, another structural, random-walks based measure, that takes into account the role of each object in the network, as well as the edge weights. **(7) Relatedness [25]**, an alternative semantic relatedness measure which considers the properties relating concepts. Unlike Lin, this measure considers all edges in the graph, therefore its computational costs are much higher.

Note that SimRank, SimRank++, and Panther do not assume an HIN model, i.e., all edges and nodes are of a single type. For these measures, we did not consider the taxonomy edges as part of the graph [2].

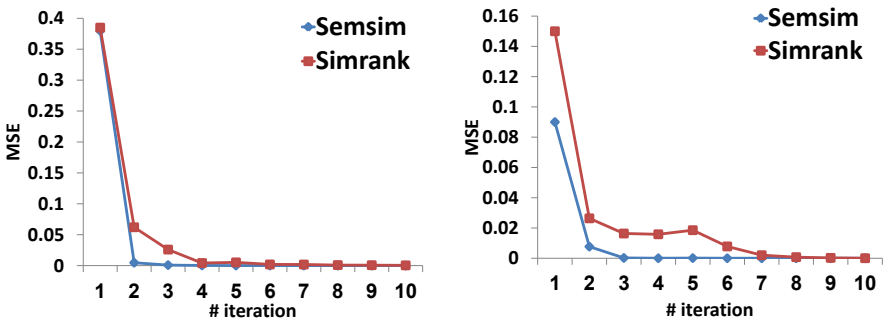
Parameters Setting To determine the decay factor and number of iterations for execution, we tested different values of c and k on the measure-quality experiments describe below.

For the decay factor, choosing smaller c values would naturally lead to smaller similarity scores, hereby faster convergence, yet our experimental results on real datasets demonstrate that setting $c = 0.8$ provided best accuracy results (it is also a typical choice for SimRank as well [21]).

We next show that a relatively small number of iterations suffices for SemSim to converge to it fix-point. From Lemma 2.9, we expect SemSim to converge as fast, or even faster, than SimRank. Indeed, as depicted in Figure 3, the differences between SemSim scores in consecutive iterations are relatively smaller than for SimRank, thus fewer iterations are required in practice. Our experiments indicate that 5 iterations are sufficient for SemSim to converge.

In all experiments (unless mentioned otherwise), the parameters for each baseline were set as follows. c and k were set to 0.8 and 5 respectively, for SemSim, SimRank and SimRank++. For the remaining algorithms we set the parameters as suggested in their corresponding papers. The meta paths considered for PathSim were all meta-paths up to length 10; we set $R = 7000$ and $t = 7$ for Panther and the taxonomy edges weight were set to 1.0, while all other weights were set to 0.4, for the Relatedness measure.

[2] We tested both possibilities and the results without the taxonomy edges were superior.



AMiner dataset Amazon dataset

Figure 3: Difference between SemSimand SimRank scores in consecutive iterations.

Table 2: Pearson's r and p-value in the WordsSim-test on Wikipedia (Wiki) and WordNet (WN).

Method	r (Wiki)	p (Wiki)	r (WN)	p (WN)
Panther	0.323	0.0376	0.206	10^{-3}
PathSim	0.293	0.0662	0.332	10^{-3}
Simrank	0.295	0.0641	0.397	10^{-4}
Simrank++	0.296	0.0644	0.395	10^{-4}
Average	0.36	0.0514	0.401	10^{-4}
Lin	0.485	0.0015	0.449	10^{-4}
Relatedness	0.510	0.0007	0.488	10^{-4}
SemSim	**0.585**	0.0001	**0.501**	10^{-4}

5.2 Measure quality

To quantitatively evaluate the adequacy of SemSim and to compare with other methods, we conducted for each dataset a dedicated experiment, by creating a typical scenario in which similarity measurements are needed, then correspondingly defining the ground truth. Interestingly, we will show that even in cases where only little semantics is available, SemSim serves as a robust similarity measure and exploits what is available, to get an edge over the competitors.

Term Relatedness Relatedness between terms is a well studied problem that requires a measure aware to both semantic and structural knowledge. To examine the adequacy of SemSim for capturing term relatedness, we used two datasets that contain relations between terms: Wikipedia and WordNet. The ground truth was defined by the WordsSim-353 test [11], a public and commonly used benchmark containing pairs of

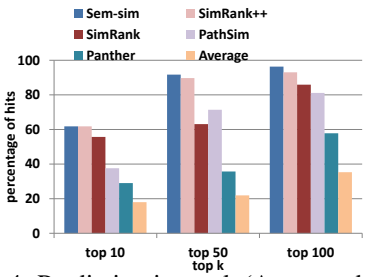


Figure 4: Prediction in top-k (Amazon dataset).

words alongside their *relatedness* scores (e.g. "computer" and "keyboard" have a relatedness score of 0.76). We then compared the scores obtained by each competitor, using the Pearson correlation measure, and reported the score and it p-value. We have removed from the benchmark pairs of words that were missing in the graph, retaining 40 pairs for Wikipedia dataset, and 342 for WordNet. Both graphs and pairs of words are available on [1].

Table 2 depicts the results for each of the tested measures. Note that other corpus-based designated methods were suggested for this task (e.g., [39]), but they require external sources beside the input graph, thus we do not include them in our benchmark. First, note that all measures which ignore semantic knowledge, demonstrate inferior results (i.e., this task highly relies on semantic knowledge). Furthermore, as determining relatedness of words is a compound task, it requires more than a simple comparison of their position in the taxonomy. Therefore Lin, an hierarchical-only measure, was found inferior to the alternatives that consider all edges (i.e., Relatedness and SemSim), yet outperforms the *Average* competitor, which emphasizes the inferiority of such an approach. Interestingly, SemSim outperforms the Relatedness measure, a designated measure for capturing the relatedness between terms. These results point out the advantage of SemSim as a comprehensive and robust similarity measure.

Link Prediction We next demonstrate how our similarity measure may be used to predict co-purchases in the Amazon dataset. We omitted 7.5K edges between co-purchased items from the original dataset, and examine how well different measures predict the missing links. Given an endpoint of a removed link, we perform a top-k search to find similar vertices by the different measures. If, for a given measure, the returned k vertices include the pair endpoint, we count a "hit", and otherwise a "miss" (a similar idea was employed to evaluate similarity search in [40]).

The results are depicted in Figure 4. For compactness, we omitted measures with particularly low uncompetitive scores. Since this task relies mostly on the structural knowledge, not surprisingly, all structural-based measures outperformed the semantic-based ones. Yet, SemSim managed to obtain a slight advantage, thanks to the semantic knowledge it uses. Note that in an analogous experiment on AMiner dataset (results omitted), we used the measures to identify multiple distinct entries representing the same author (e.g, `Susan B. Davidson`; `Susan Davidson`), PathSim succeed better compared to SimRank++, due to the additional semantic knowledge that was available in which SimRank++ ignores and PathSim considers. Yet, again, SemSim outperformed both. In the latter, the reason is that the semantic knowledge only provides information about the field-of-interests and not on the authors (recall Example 2.6). The *Average* approach demonstrated inferior results in both experiments, in comparison to other competitors.

In a real life scenario, both link prediction and entity resolution engines may use additional information to what is available in our datasets (e.g. user profiles, reviews, rating, etc.) along with a variety of data mining techniques. Yet our experimental results demonstrate that SemSim makes a substantially exhaustive exploitation of the available structural and semantic information.

A closer look To better understand the behavior of SemSim compared to the alternatives, we took a closer look at the AMiner experiment mentioned above, this time examining the top-5 most similar researchers selected for various authors. As an illustrative example, Table 3 shows the result obtained for the author Tova Milo.

SemSim	Simrank++	Simrank	PathSim	Panther
Daniel Deutch	Daniel Deutch	Ohad Greenshpan	Rajeev Rastogi	Daniel Deutch
Neoklis Polyzotis	Ohad Greenshpan	Daniel Deutch	Michael Benedikt	Neoklis Polyzotis
Serge Abiteboul	Elad Verbin	Elad Verbin	Werner Nutt	George Candea
Susan Davidson	Magdalini Eirinaki	George Candea	Dan Olteanu	Radek Vingralek
Victor Vianu	Neoklis Polyzotis	Radek Vingralek	Susan Davidson	Ohad Greenshpan

Table 3: Top 5 similar authors to Tova Milo.

Dataset	G^2	$G^2_\theta, \theta = 0.95$	$G^2_\theta, \theta = 0.9$
AMiner	n = 60M	n = 14K	n = 9K
	m =2.2B	m =39M	m = 7.8M
Wikipedia	n = 22M	n = 10K	n = 6K
	m =10.2B	m = 23.5M	m = 4.7M

Table 4: The size of G^2 and G^2_θ for $\theta = 0.9$ (top $\approx$ 5K) or $\theta = 0.95$ (top $\approx$ 1K), in different datasets.

One can see that all measures except PathSim favors past collaborators of Milo. PathSim returns a set of authors concentrated on a small number of venues and fields of interest shared with Milo, considering common collaborators and only fields of interest that are exact match (no synonyms or related topics); SimRank tends to return past collaborators, but ignored the amount of collaborations (i.e., weights); SimRank++ and Panther prefer authors that share similar collaborators, this time considering weights, whereas SemSim favors collaborators with semantically related fields of interest, which make sense in this context and captures the desired similarity in this network.

5.3 Performance evaluation

We next examine the optimization techniques suggested in Sections 3.2 and 4 and analyze their performances. We start by reporting the size of the pair-node graph G^2, with and without semantic-based pruning. Then, we analyze the running times and accuracy of our approximation algorithm.

G^2 pruning In Section 3.2, we suggested constructing a pruned version of G^2, G^2_θ. Table 4 presents the size of G^2_θ compared to G^2, for different choices of θ. While G^2_θ is indeed significantly smaller, note that this technique is still relevant only for not too large graph, hence, this is the set of graphs we considered here. Larger graphs are handled below through approximated computation.

Considering Table 4, one can see that almost 99.9% of the edges and nodes were omitted in all cases. The choice of high θ values ensures that only the most "interesting" pairs would be considered (top 1K or 5K pairs), i.e., only comparing subset of authors (in AMiner) or semantically related terms (in Wikipedia). This compact representation is useful in cases only objects whose similarity score is above a given threshold are of interest.

Approximated SemSim When even our reduced graph might still be too large, one can employ the approximation algorithm, directly on G. We next report the accuracy and running times of our approximated framework. We analyze the sensitivity of its parameters, number of walks and walk's length, from two aspects: (i) accuracy and (ii) running times, on AMiner and Amazon datasets.

Dataset	Exp. Max Error	Measured Max Error	Mean Error
Amazon	0.028	U: 0.020 W: 0.018	U: 10^{-4} W: 10^{-4}
AMiner	0.050	U: 0.044 W: 0.041	U: 10^{-3} W: 10^{-3}

Table 5: Accuracy of approximation, using 2 sampling strategies: Uniform (U) and Weighted (W).

Accuracy As a ground truth, to which we compare the approximated scores, we use the similarity scores computed by the baseline algorithm (see Section 2.4), setting the number of iterations to 10. We run this experiment only on the smaller versions of the graphs due to the non-scalability of the baseline naive algorithm.

The parameters k and d (maximal ratio between edge weights and maximal in-degree) are determined by the dataset. Their values for Amazon and AMiner datasets are $k = 1.7, d = 2.7$K and $k = 2, d = 72$, respectively. We next analyze the effect of varying n_w and t, where the decay factor c is set to 0.8, as explained in Section 5.1. Of course, there is a trade-off between running times and accuracy, i.e., increasing n_w and t would decrease the error, yet increase the running times. According to our experiments, a reasonable choice of the parameters, which obtained sufficient error bound and short running times is to set: $n_w = 50, t = 15$ [3].

Table 5 depicts the maximal and mean error incurred by each sampling strategy, uniform and weighted, in 1K randomly picked node-pairs similarity computations over 10 different runs, where each run rebuilds the index from scratch. Observe that the mean error is considerably smaller than the stipulated theoretical error bound, which is consistent with our analysis for the expected error in the worst-case.

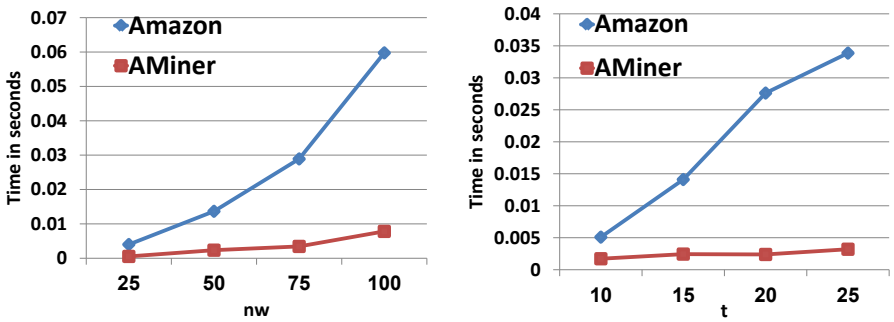
Running time We examine the performance of our suggested method as a function of its parameters: the number of walks n_w and the truncation point t (k and d are fixed in each graph), on the two large versions of Amazon and AMiner datasets. We start by analyzing the query running times then consider the preprocessing phase.

Figure 5 depicts the average running times of 1K randomly picked node-pairs for single-pair queries using varying values of n_w and t, while maintaining a cache storing up to $|V|$ last LCA calculations. In all experiments the average running time was less than 0.02 seconds (and 0.06 at most). Not surprisingly, since we consider all coupled random-walks (i.e., n_w^2 walks) the effect, on running times, of increasing n_w is greater than that of increasing t.

The differences between the two datasets is directly related to the taxonomy size, more concretely, the LCA calculations. In Amazon dataset, where the taxonomy contains 2.5M edges, finding the LCA takes longer than in AMiner, where the taxonomy contains only 350K edges. Importantly, most of the time (around 80%) was spent on the online semantic similarity computations, i.e, the LCAs computation. Yet, we note that the cashing was very effective here as without it the computation would take up to 5 times longer. In comparison, using Monte-Carlo method to compute SimRank with the same parameters was approximately 7 times faster, but, as shown in the previous experiments yields inferior similarity accuracy.

The offline phase consists of two parts: sampling the random walks and pre-processing the taxonomy. Note that the first part allows a straightforward parallelization of indexing: the computation of independent index databases can be performed on up to n_w

[3] We do not report the results for different choices of parameters for lack of space, but the same effect was measured: in practice, the error is much smaller than the error bound.



(a) t is set to 15 (b) n_w is set to 50

Figure 5: Single-pair queries average running time.

different machines. In this experiment we did not parallelize the indexing, and consider the computation on a single machine. According to our experiments (for different values of n_w and t), both parameters have a linear effect on the pre-processing time, and in average, this phase took approximately 2.5 minutes. As for pre-processing the taxonomy, for Amazon dataset, where the taxonomy is much larger, this phase took approximately 7 minutes and for AMiner dataset approximately 1.5 minutes.

Finally, the affect that taxonomy size has on running times (in both offline and online phases), opens up an opportunity for further optimizations, such as approximated Lin scores, which we leave for future work.

6 Related Work

Estimating vertex similarity has been studied in various contexts and with different goals in mind. For example, recommender systems leverage similarities between users/items in order to recommend items [27]; in the context of sponsored search, where paid advertisements, relevant to a userâĂŹs query, are shown, similarity measures are essential [4]; and in social networks, friend recommendations are computed using techniques of link prediction, which rely on the analysis of the network structure [7, 34]. The integration of semantic and structural similarities allows the system to make inferences based on the underlying reasons for which a user/item may be relevant in a particular scenario. Moreover, in cases where little or no structural information is available (such as in the case of newly added items/user), the system can still use the semantics to provide reasonable recommendations.

Generally, there are two basic principles to quantify the structural similarity between two vertices: (i) in direct relation to the number of common neighbors in the network, based on the transitivity of similarity [17, 36, 42] and (ii) in direct relation to the similarity of their structural roles [35, 40]. We adopted SimRank because of its generality, simplicity and wide range of optimizations. Due to its computational complexity, efficient calculation and approximation of SimRank has been studied intensively. Following [41], those algorithms can be classified into 3 classes: (1) iterative methods, [24], (2) non-iterative methods, which solve a linear system [21], and (3) random-walks based methods [37, 18]. As previously discussed, a large number of these works can be applied to our setting.

Collaborative filtering is a method of making automatic predictions about the interests of users. The underlying assumption of this approach is that two similar users are more likely to share similar preferences. Typically, similarity is quantified by com-

paring a feature vector of ratings, possibly exploiting semantic knowledge [2, 27]. Yet, unlike SemSim, this approach does not consider the network structure. Such feature-based measures may be incorporated into our formula as weights, in a similar manner to our incorporation of semantics.

Much efforts were devoted to quantify the semantic similarity or relatedness in ontologies, a special case of HIN, especially with the increasing interest in the Semantic Web and the popularization of ontologies. Generally, there are two main classes of semantic similarity measures: (1) IC based measures [29, 25, 23] and (2) feature-based measures [22, 39]. The latter usually involves external sources beside the input graph in the computation. We chose to adopt the Lin measure due to its simplicity, common usage and most importantly, its efficiency. Although, replacing Lin with a different semantic measure is possible, it would most likely lead to a much higher computational cost.

Another important aspect of similarity measures in networks is the underlying data model. Commonly, two models were considered: (i) the homogeneous model [17, 40], where nodes and edges belong to a single type and (ii) the heterogeneous model [4, 35, 25, 42], where nodes and edges belong to a set of types. With the advent of large-scale HINs [32, 38], such as bibliographic and social media networks, we adopted the HIN model due to its straightforward representation of all available data.

Despite much research on the topic, the problem of combining both sources of knowledge and quantifying a unified similarity measure, remains largely unsolved. Previous works have suggested combining structural and semantic knowledge to form a single similarity score [15, 14, 27, 13], yet, while they all make a notable contribution, each of the proposed solutions focuses on a specific task and does not consider the network structure, i.e., the relations between objects, considering instead only the structure of the object itself (e.g., [15] considered the structure of XML pages). Furthermore, scalability issues were not addressed.

A related research area is *Ontology matching*. The matching operation determines an alignment for a pair of ontologies [33, 10]. This task incorporates both semantic and structural knowledge. While some of these methods resemble ours in measuring the similarity of nodes based on common neighbors and using taxonomical data, their goal is different: they aim to identify equivalent representations of the same entity. Thus, in contrast to our work, quantifying the similarity of distinct entities is not targeted. Moreover, our data model, which involves edge weights, is not accounted by the proposed measures.

Lastly, another related line of work attempts to determine the relatedness or importantness between/of terms [25, 5, 4, 6]. While this line of work, which also considers both structural and semantic properties, makes considerable advances, their aim is, once again, different: the relatedness measures are designated measures that only allow taking into account functional relations between terms, while the importantness measures aim to find all relevant terms to a given query. Hence, these measures are not applicable in the general case, were the similarity between arbitrarily objects is desired.

7 Conclusion and Future work

In this paper we introduce SemSim, a novel similarity measure, which extends SimRank, by integrating semantic knowledge into the evaluation. As pointed out in Section 5, in many cases, existing techniques seem ill-suited to capturing the desired similarity, that relies on both structure and semantics, and a deeper integration between the two

concepts is needed. In this work we make initial steps in this direction, presenting a robust measure for networks, that takes into account all this relevant data. We presented a dedicated pruning techniques for SemSim, as well as an efficient probabilistic framework, anchored in a careful modification of the underlying model of SimRank, that preserves the necessary properties for the application of numerous optimizations previously suggested for SimRank. We further demonstrated the usefulness of our measure by showing how it captures desired notions of similarity in real datasets. Our experimental study indicates the efficiency of our algorithm and its scalable manner.

An interesting direction for future work is to extend SemSim with parameters that allow for tailoring to specific scenarios. A natural staring point would be drawing inspiration from a plethora of SimRank variations. For example, [44] considered a probabilistic graph setting. Furthermore, in this work we focused on a single-pair queries. We are currently pursuing an extension of our work to top-k and similarity-join queries, inspired by [19, 43].

References

[1] Term relatedness. `https://github.com/TAU-DB/SemSim/blob/master/Relatedness_Test.txt`.

[2] C. C. Aggarwal and S. Y. Philip. Semantic based collaborative filtering, Nov. 26 2002. US Patent 6,487,539.

[3] AMiner. `https://aminer.org/data`.

[4] I. Antonellis, H. G. Molina, and C. C. Chang. Simrank++: query rewriting through link analysis of the click graph. *PVLDB*, 2008.

[5] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, 2004.

[6] R. Bonaque, B. Cautis, F. Goasdoué, and I. Manolescu. Toward social, structured and semantic search. In *SDSW*, 2014.

[7] H. Chen, X. Li, and Z. Huang. Link prediction approach to collaborative filtering. In *JCDL*, 2005.

[8] J. Cheng, S. Huang, H. Wu, and A. W.-C. Fu. Tf-label: a topological-folding labeling scheme for reachability querying in a large graph. In *SIGMOD*, 2013.

[9] DBpedia. `http://dbpedia.org/About`.

[10] J. Euzenat, P. Shvaiko, et al. *Ontology matching*. Springer, 2007.

[11] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *WWW*, 2001.

[12] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, 2005.

[13] P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting hierarchical domain structure to compute similarity. *TOIS*, 2003.

[14] A. Günay and P. Yolum. Structural and semantic similarity metrics for web service matchmaking. In *E-Commerce and Web Technologies*. 2007.

[15] R. Guzman, I. Dongo, and R. T. Herrera. Structural and semantic similarity for xml comparison. In *MEDES*, 2013.

[16] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *siam Journal on Computing*, 1984.

[17] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *SIGKDD*, 2002.

[18] M. Kusumoto, T. Maehara, and K.-i. Kawarabayashi. Scalable similarity search for simrank. In *SIGMOD*, 2014.

[19] P. Lee, L. V. Lakshmanan, and J. X. Yu. On top-k structural similarity search. In *ICDE*, 2012.

[20] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014.

[21] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of simrank for static and dynamic information networks. In *EDBT*, 2010.

[22] J. Liang, D. Ajwani, P. K. Nicholson, A. Sala, and S. Parthasarathy. What links alice and bob?: Matching and ranking semantic patterns in heterogeneous networks. In *WWW*, 2016.

[23] D. Lin. An information-theoretic definition of similarity. In *ICML*, 1998.

[24] D. Lizorkin, P. Velikhov, M. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for simrank computation. *VLDB*, 2008.

[25] L. Mazuel and N. Sabouret. Semantic relatedness measure using object properties in an ontology. In *ISWC*, 2008.

[26] G. A. Miller. WordNet: a lexical database for English. *ACM*, 1995.

[27] B. Mobasher, X. Jin, and Y. Zhou. Semantically enhanced collaborative filtering on the web. In *EWMF*. 2004.

[28] N. F. Noy, M. A. Musen, et al. Algorithm and tool for automated ontology merging and alignment. In *AAAI*, 2000.

[29] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[30] N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in wordnet. In *ECAI*, 2004.

[31] Y. Shao, B. Cui, L. Chen, M. Liu, and X. Xie. An efficient similarity search framework for simrank over large dynamic graphs. *VLDB*, 2015.

[32] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip. A survey of heterogeneous information network analysis. *TKDE*, 2017.

[33] P. Shvaiko and J. Euzenat. Ontology matching: state of the art and future challenges. *TKDE*, 2013.

[34] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *ASONAM*, 2011.

[35] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 2011.

[36] W. Tao, M. Yu, and G. Li. Efficient top-k simrank-based similarity join. *PVLDB*, 2014.

[37] B. Tian and X. Xiao. Sling: A near-optimal index structure for simrank. In *SIGMOD*, 2016.

[38] C. Wang, Y. Sun, Y. Song, J. Han, Y. Song, L. Wang, and M. Zhang. Relsim: Relation similarity search in schema-rich heterogeneous information networks. In *SDM*, 2016.

[39] W.-t. Yih and V. Qazvinian. Measuring word relatedness using heterogeneous vector space models. In *NAACL HLT*, 2012.

[40] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li. Panther: Fast top-k similarity search on large networks. In *SIGKDD*, 2015.

[41] Z. Zhang, Y. Shao, B. Cui, and C. Zhang. An experimental evaluation of simrank-based similarity search algorithms. *PVLDB*, 2017.

[42] P. Zhao, J. Han, and Y. Sun. P-rank: a comprehensive structural similarity measure over information networks. In *CIKM*, 2009.

[43] W. Zheng, L. Zou, Y. Feng, L. Chen, and D. Zhao. Efficient simrank-based similarity join over large graphs. *VLDB*, 2013.

[44] R. Zhu, Z. Zou, and J. Li. Simrank computation on uncertain graphs. In *ICDE*, 2016.

Entity	IC value
Thing	0.01
Author	0.9
Computer Science (CS)	0.1
Alice, Bob, Carol	1.0
Data Management (DM)	0.5
Information Systems (IS)	0.2
Crowdsourcing	0.9
Databases (DB)	0.2
Information Management (IM)	0.5

Table 6: IC values for Figure 1 entities.

A Full computation for Example 2.6

We next provide full computation to the example introduce in Section 2. The decay factor used for both SimRank and SemSim was 0.8 (as suggested in Section 5.1). Additionally, The missing edge weights were set to 1, and the IC values (depicted in Table 6) were computed using Seco formula [30], on the domain ontology for AMiner dataset (which include a taxonomy of computer science related fields of research). First, computing pair-wise SimRank scores, (in this small graph 3 iterations were sufficient) one would obtain $simrank(\texttt{Alice,Bob}) = simrank(\texttt{Bob,Carol}) \approx 0.21$. As for Lin, since all authors nodes are leafs in the taxonomy, their corresponding IC values are all 1, thus $Lin(\texttt{Alice,Bob}) = Lin(\texttt{Bob,Carol}) = 0.9$. Using the IC values above, one would obtain $Lin(\texttt{Crowdsourcing,DB}) = 0.9$ and $Lin(\texttt{DB,IM}) = 0.18$.

Next, we briefly overview SemSim computation (with the same parameters). At the first step, since $\texttt{Bob} \neq \texttt{Alice} \neq \texttt{Carol}$, $R_0(\texttt{Alice,Bob}) = R_0(\texttt{Bob,Carol}) = 0$. Iteratively, at the next step, since all three share two common neighbors, `Author` and `Database`, we get $R_1(\texttt{Alice,Bob}) = R_1(\texttt{Bob,Carol}) = 0.09$. At the next step, the semantic similarity of common neighbors propagates into the computation, thus, $s(\texttt{Alice,Bob}) = 0.124$, while $s(\texttt{Bob,Carol}) = 0.120$. Finally, in the last iteration we get the final scores of 0.133 and 0.128 for Alice and Carol, respectively.

B Properties of Sem-Sim

We next prove the following properties holds for SemSim.

Theorem B.1. *The iterative* SemSim *equations (shown in Equation (3) and Equation (4)) have the following properties:*
1.(Symmetry) $\forall a, b \in V R_k(a,b) = R_k(b,a)$
2.(Maximum self similarity) $\forall a \in V : R_k(a,a) = 1$
3.(Monotonically) $0 \leq R_k(a,b) \leq R_{k+1}(a,b) \leq 1$
4.(Existence) *The solution to the iterative equations always exists and converges to a fixed point,* $s(\cdot,\cdot)$*, which is the theoretical solution to the recursive equations.*
5.(Uniqueness) *the solution to the iterative equations is unique when* $c \neq 1$.

Proof. According to Equations (3) and (4), it is obvious that the **Symmetry** and **Maximum self similarity** requirements from SemSimholds, i.e, $\forall a, b \in V R_k(a,b) = R_k(b,a)$ and $R_k(a,a) = 1$ for all k.
3.(Monotonicity) If $a = b, R_0(a,b) = R_1(a,b) = ... = 1$, so it is obvious that the monotonicity property holds. LetâĂŹs consider $a \neq b$. According to Equation (3),

$R_0(a,b) = 0$. Base on Equation (4),$0 \le R_1(a,b) \le 1$. So, $0 \le R_0(a,b) \le R_1(a,b) \le 1$. LetâĂŹs assume that for all $k, 0 \le R_{k-1}(a,b) \le R_k(a,b) \le 1$, then

$$R_{k+1}(a,b) - R_k(a,b) = \frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot [R_k(a,b) - R_{k-1}(a,b)]$$

Based on the assumption we have $R_k(a,b) - R_{k-1}(a,b) \ge 0, \forall a,b \in V$, so the left hand side $R_{k+1}(a,b) - R_k(a,b) \ge 0$ holds. By induction, we draw the conclusion that for any $k, R_k \le R_{k+1}$. And based on the assumption, $0 \le R_k(a,b) \le 1$, so

$$R_k(a,b) = \frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot (R_k(a,b) \le \frac{c \cdot Lin(a,b)}{\sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} 1 \cdot 1} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} 1 \cdot 1 \cdot 1 = c \cdot Lin(a,b)$$

so, $R_{k+1}(a,b) \le c \cdot Lin(a,b) \le 1$. By induction, we know that for any $k, 0 \le R_k(a,b) \le 1$.
4.(Existence) According to the previous proof, $\forall a,b \in V, R_k(a,b)$ is bounded and non-decreasing as k increases. By the Completeness Axiom of calculus, each sequence $R_k(a,b)$ converges to a limit $R(a,b) \in [0,1]$. Note $\lim_{k\to\infty} R_k(a,b) = \lim_{k\to\infty} R_{k+1}(a,b) = R(a,b)$, so we have:

$$R(a,b) = \lim_{k\to\infty} R_{k+1}(a,b) = \frac{c \cdot Lin(a,b)}{N_I} \lim_{k\to\infty} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot R_k(a,b) = \frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot \lim_{k\to\infty} R_k(a,b) = \frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot R(a,b)$$

Note that the limit of $R_k(\cdot,\cdot)$, with respect to k, right satisfies the recursive similarity equation, shown in the Definition 2.5.
5.(Uniqueness) Suppose $s_1(*,*)$ and $s_2(*,*)$ are two solutions to the n^2 iterative similarity equations. For any nodes $x, y \in V$, let $\delta(x,y) = s_1(x,y) - s_2(x,y)$ be their difference. Let $M = max_{(x,y)}|\delta(a,b)|$ be the maximum absolute value of any difference. We need to show that $M = 0$. Let $|\delta(x,y)| = M$ for some $a,b \in G$. It is obvious that $M = 0$ if $a = b$. Otherwise,

$$\delta(a,b) = s_1(a,b) - s_2(a,b) = \frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b)[s_1(I_i(a),I_j(b)) - s_2(I_i(a),I_j(b))]$$

Thus,

$$M = |\delta(a,b)| =$$

$$|\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot \delta(I_i(a), I_j(b))| \leq$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot |\delta(I_i(a), I_j(b))| \leq$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot M = c \cdot Lin(a,b) \cdot M$$

So, $M = 0$ when $c \neq 1$ and $\forall a, b \in VLin(a,b) \neq 0$ (Observation 2.4). □

We next provide a prove for Lemma 2.9.

Proof. If $a = b$ or $I(a) = \emptyset$ or $I(b) = \emptyset$, according to equations (3) and (4),

$$R_{k+1}(a,b) - R_k(a,b) = 0 < Lin(a,b) \cdot c^{k+1}$$

For the general case of $a \neq b$, $I(a) \neq \emptyset$ and $I(b) \neq \emptyset$, the proof is organized by mathematical induction.
(Induction Basis.)($k = 0$)

$$R_1(a,b) - R_0(a,b) = R_1(a,b) =$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot R_0(a,b)$$

Since $0 \leq R_0(I_i(a), I_j(b)) \leq 1$,

$$0 \leq R_1(a,b) - R_0(a,b) \leq$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot 1 = c \cdot Lin(a,b)$$

(Inductive step.) Provided that the lemma holds for a given integer k, ($k > 0$) for all vertex pairs, i.e.,

$$0 \leq R_k(a,b) - R_{k-1}(a,b) \leq Lin(a,b) \cdot c^k$$

let us prove the lemma holds for $k + 1$ as well.

$$R_{k+1}(a,b) - R_k(a,b) =$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot R_k(a,b) -$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot R_{k-1}(a,b) =$$

$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot [R_k(a,b) - R_{k-1}(a,b)]$$

Based on the assumption we get:

$$R_{k+1}(a,b) - R_k(a,b) \leq$$
$$\frac{c \cdot Lin(a,b)}{N_I} \sum_{i}^{|I(a)|} \sum_{j}^{|I(b)|} W(I_i(a),a) \cdot W(I_j(b),b) \cdot c^k =$$
$$c \cdot Lin(a,b) \cdot c^k = Lin(a,b) \cdot c^{k+1}.$$

Therefore, according to the induction method, we can conclude that the Lemma holds. □