Toward Representation Independent Similarity Search Over Graphs

Yodsawalai Chodpathumwan
University of Illinois
ychodpa2@illinois.edu

Arash Termehchy
Oregon State University
termehca@eecs.oregonstate.edu

Yizhou Sun
Northeastern University
yzsun@ccs.neu.edu

Amirhossein Aleyasin
University of Illinois
aleyase2@illinois.edu

Jose Picado
Oregon State University
picadolj@eecs.oregonstate.edu

ABSTRACT

Finding similar entities over data graphs is an important problem with many applications. Current similarity search algorithms use intuitively appealing heuristics that leverage the link information in the data graph to quantify the degree of similarity between its entities. In this paper, using examples from real-world data sets, we show that people represent the same information using data graphs with different shapes. We argue that in order for a similarity search algorithm to be usable and effective, it should be **representation independent**: it should return essentially the same answers for a query over different graphs that represent the same information. We formalize this property and show that the outcome of current similarity search algorithms depend highly on data representation. Hence, they may be effective on some datasets and ineffective over others. We also perform an empirical study and analyze the sensitivity of current methods against changes in data representation. Our results indicate that the output of these algorithms are highly affected by changes in data representation.

1. INTRODUCTION

Finding similar entities in graph databases is an important and popular information need with applications in many domains, such as Web, social networks, and scientific data [3, 4, 7]. In the last decade, as data sets became more heterogeneous and complex, the difficulty of this task multiplied [3, 4]. Since the properties of *similar* entities cannot be precisely defined in a query, current similarity search algorithms use intuitively appealing heuristics that leverage information about the links between entities to choose, from among all possible answers, those that are most similar to the input entity. These heuristics normally implement the idea that similar entities are *strongly* connected via links in the database.

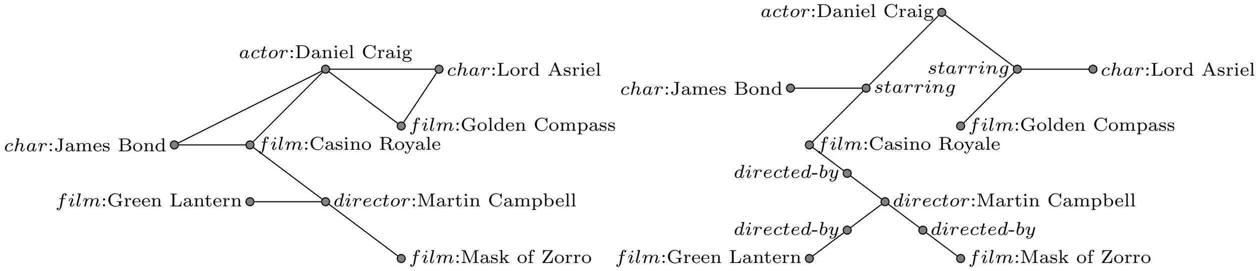
For example, a user may want to find similar movies in IMDb (*imdb.com*), which contains information about movies, actors and directors (see fragment in Figure 1a). With no traditional query available, Data exploration algorithms leverage the concept of *proximity* to find strong relationships in the data [3, 4, 6]. They quantify the proximity between entities by, among other heuristics, computing the length of the shortest path between them, computing their PageRank score in the node-pair graph (SimRank), or the probability that a random walk with restart (RWR) leads from one to the other. For instance, if a user asks for movies that are similar to *Casino Royale*, since the RWR score and SimRank of *Golden Compass* (RWR-score = 0.063, SimRank-score = 0.196) are larger than that of *Mask of Zorro* or *Green Lantern* (RWR-score = 0.041, SimRank-score = 0.141) , an algorithm might rank them in that order and report them as being similar to *Casino Royale*.

The power of similarity search algorithms remains out of the reach of most users, however, as today's similarity search algorithms and tools are usable only by trained data analysts who can predict which algorithms are likely to be effective for particular queries and datasets, or who are able to customize these algorithms to satisfy their information needs over a new dataset. To see why, consider the excerpts of Freebase (*freebase.com*) in Figure 1b. Freebase contains facts about entities and their relationships, harvested mainly from Wikipedia. IMDb and Freebase use different representations to express the same relationship between *Casino Royale* and *Golden Compass*. If a data exploration algorithm uses a proximity-based heuristic, such as RWR or SimRank, to pick out closely related entities, it will find *Casino Royale* more closely related to *Mask of Zorro* and *Green Lantern* (RWR-score = 0.024, SimRank-score = 0.070) than to *Golden Compass* (RWR-score = 0.016, SimRank-Score = 0.064) in Figure 1b.

Furthermore, an algorithm that leverages edge distances to compute proximity will also deliver a different ranking between the two databases in Figure 1 because *Casino Royale* and *Golden Compass* are relatively far apart in Figure 1b than in Figure 1a. Therefore, a data exploration algorithm that successfully discovers a strong relationship between *Casino Royale* and *Golden Compass* in IMDb may not find the same relationship interesting over Freebase, without further customization.

More generally, there is no canonical or ideal structure for representing a particular set of content. For instance, many

GRADES '14, June 22 - 27 2014, Snowbird, UT, USA
Copyright 2014 ACM 978-1-4503-2982-8/14/06 ...$15.00.
http://dx.doi.org/10.1145/2621934.2621946



(a) IMDb (b) Freebase

Figure 1: Fragments of IMDb (*imdb.com*) and Freebase (*freebase.com*) databases.

researchers use DBLP (*dblp.uni-trier.de*), a publicly available bibliographical database about computer science publications, to validate and evaluate their algorithms. We have observed that different researchers often represent DBLP data differently. Figure 2 shows two different ways DBLP data was organized in two papers in premier conferences in data management and mining [4, 7]. However, one may change *term* to *year* in [4] and get two schemas of DBLP that represents the same information.

Expert customization may be well worth the expense for data sets as popular as IMDb and Freebase, but will not be affordable for the masses of users who would like to draw insights from data in the long tail, i.e., originating from less popular sources, or created by merging and mashing data from multiple sources. For similar reasons, current similarity search algorithms will not be well suited for Big Data, which is inherently heterogeneous.

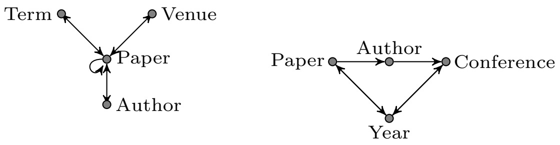
To the best of our knowledge, the problem of representation independence has not previously been defined and explored for algorithms over graph data. Our contributions in this paper are as follows.

- We introduce and formally define the property of generality for similarity search algorithms over graph databases. We introduce a family of information preserving transformations over graph data and define the property of generality for similarity search algorithms as robustness under these transformations.
- We empirically study the generality of some widely used similarity search algorithms. Our results indicate that modifying representation of the data considerably affects the results of these algorithms.

This paper is organized as follows. Section 2 describes the related works, Section 3 formally defines the property of generality for similarity search algorithms. Section 4 contains our empirical results, and Section 5 concludes the paper.

2. RELATED WORK

Database researchers envisioned the desirable property of *logical data independence* from the early days [1]. This property requires a database management system to return the same answers for a relational query no matter what schema is chosen for the data. Our recent work on *design-independent* query answering algorithms extended the concept of logical data independence to apply to keyword queries over XML data [5]. This paper, however, introduces and studies the concept of generality for similarity search algorithms over graph databases. Because graph data model is more complex than XML data model, one cannot simply extend the notion of design independence from XML for graph databases. Further, the task of entity similarity search has a different semantic than the one of keyword search and therefore requires different algorithms. Current entity similarity search algorithms also differ from keyword query methods.



(a) DBLP schema from [4]. (b) DBLP schema from [7].

Figure 2: Different schemas for DBLP data set.

3. INFORMATION EQUIVALENT GRAPHS

We model a database as a graph $D = (V, E, \mathcal{L}, \mathcal{A})$, where V is the set of nodes in the graph, E is the set of edges between members of V, and $\mathcal{L}$ and $\mathcal{A}$ respectively assign a label and an optional data value (e.g., a string) to all the members of V. Figure 1a and 1b show fragments of two graph databases. In some databases, such as graph representations of RDF, some nodes may not have any value. A **query** for an entity similarity search algorithm over database $D(V, E, L, A)$ is defined as $q = (v)$ where $v \in V$ and asks for similar nodes to v in the database. We call v a *query node*. An example of a query over graph database fragment shown in Figure 1a is (*film: Mask of Zorro*). A similarity search algorithm returns a ranked list of nodes as the answer for a query. We denote the result of query q over D as $q(D)$.

Database transformations have been used to formally capture the relationships between different choices of structure in terms of the amount of information they can contain [2]. A **transformation** T over database D is a function that modifies D to generate a new database $T(D)$. Transformation T is **invertible** if and only if there is a transformation T^{-1} such that $T^{-1}(T(D)) = D$. That is, we can reconstruct the information available in the original database given the transformed database. If there is an invertible transformation from database D_1 to D_2 and from D_2 to D_1, D_1 and D_2 essentially represent the same information [2]. We call such databases *information equivalent*.

We further restrict the properties of an invertible transformation to satisfy the requirements and settings of similarity search algorithms. An invertible transformation may introduce or eliminate finite number of data items in a way that is recoverable using its inverse [2]. For example, one can define transformation U and its inverse U^{-1} over a database as in Figure 1a such that U replaces all occurrences of *Daniel Craig* with *Alan Dan* and U^{-1} replaces all occurrences of *Alan Dan* with *Daniel Craig*. However, to get the same answers for query s: *Find actors similar to Daniel Craig* over the original and transformed databases, we must modify s to *Find actors similar to Alan Dan*. We want our similarity algorithms to be sufficiently general that users do not have to drastically modify their queries over different representations of the same information. For that reason, we consider only the transformations, which leave the values stored in the database intact. Since similarity search algorithms often treat the labels of nodes as their types, it is natural for an information preserving transformation T to map nodes of the same type in original database D to nodes from the same types in the transformed database $T(D)$.

Definition 1. An invertible transformation T that maps database $D_1(V_1, E_1, \mathcal{L}_1, \mathcal{A}_1)$ to database $D_2(V_2, E_2, \mathcal{L}_2, \mathcal{A}_2)$ is *information preserving* iff there is a total and surjective relation R between nodes with values in D_1 and D_2 such that

- Nodes e and $R(e)$ have equal values.
- If we have $\mathcal{L}_1(e_1) = \mathcal{L}_1(e_2)$, then $\mathcal{L}_2(R(e_1)) = \mathcal{L}_2(R(e_2))$.

Transformations that add or remove duplicate values from databases and preserve their information contents, e.g. database normalization and denormalization, are very frequently used in data processing. Thus, Definition 3 allows for the (de-)duplications of nodes in a database. Figures 3a(a) and 3a(b) illustrate fragments of two representations of DBLP database that are organized according to the schemas in Figures 2(a) and 2(b), respectively. The transformation between two databases duplicates the *conference* nodes in Figure 3a(a). It is easy for the query interface to check and remove duplicates in their answers.

If an information preserving transformation (de-)duplicates some nodes in the database, it is hard to translate the queries that ask about similar entities to (de-)duplicated nodes from the original to the transformed database. For instance, there is only one conference node with value $PVLDB$ in Figure 3a(a) but there are more than one such node in Figure 3a(b). Hence, it is not clear how to translate query *conference:PVLDB* over Figure 3a(a) to a query over Figure 3a(b). A similarity search algorithm may deliver a different result for each *conference:PVLDB* node over Figure 3a(b). Hence, we further limit information preserving transformations to be able to translate similarity queries over equivalent databases. We show all nodes with values in database D whose labels belong to set $F \subseteq \mathcal{L}(D)$ as $F(D)$.

Definition 2. An information preserving transformation T that maps $D_1(V_1, E_1, \mathcal{L}_1, \mathcal{A}_1)$ to $D_2(V_2, E_2, \mathcal{L}_2, \mathcal{A}_2)$ is *query preserving* w.r.t. $L_1 \subseteq \mathcal{L}_1(D_1)$ and $L_2 \subseteq \mathcal{L}_2(D_2)$ iff there is a bijective mapping $M : L_1(D_1) \rightarrow L_2(D_2)$ such that

- Nodes e and $M(e)$ have equal values.
- If $\mathcal{L}_1(e_1) = \mathcal{L}_1(e_2)$, we have $\mathcal{L}_2(M(e_1)) = \mathcal{L}_2(M(e_2))$.

Generally, users are mainly interested in only a subset of labels for their queries. For instance, users of a movie database are mostly interested in finding similar actors or movies rather than similar countries. A query preserving transformation T w.r.t. L_1 and L_2, over database D, provides a bijective mapping between all potential queries in D and $T(D)$ that preserves value and label information. Hence, it is possible to submit essentially same queries over databases D and $T(D)$. By the abuse of notation, we denote the query over $T(D)$ that is mapped to q as $T(q)$. $D_1(V_1, E_1, \mathcal{L}_1, \mathcal{A}_1)$ and $D_2(V_2, E_2, \mathcal{L}_2, \mathcal{A}_2)$ are *query equivalent* (equivalent for short) w.r.t. $L_1 \subseteq \mathcal{L}_1(D_1)$ and $L_2 \subseteq \mathcal{L}_2(D_2)$ iff there is a query preserving transformation from D_1 to D_2 w.r.t. L_1 and L_2 and a query preserving transformation from D_2 to D_1 w.r.t. L_2 and L_1. Next, we define this notion of generality formally as follows.

Definition 3. A similarity search algorithm A is *general* iff given query q and $T(q)$ over equivalent database D and $T(D)$, respectively, there is one-to-one mapping M between $q(D)$ and $T(q)(T(D))$ where

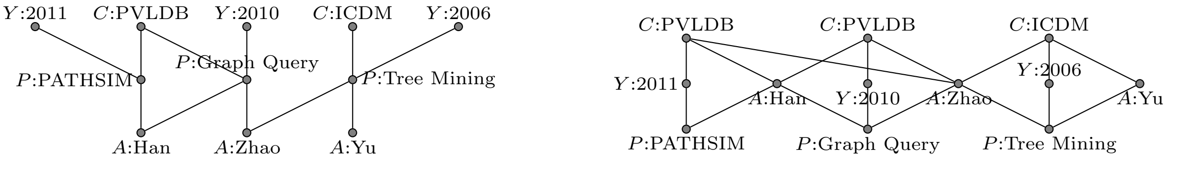
- If $e \in q(D)$ and $M(e) \in T(q)(T(D))$, we have $M(e) = T(e)$.
- Node $e \in q(D)$ appears before node $f \in q(D)$ iff node $M(e) \in T(q)(T(D))$ appears before node $M(f) \in T(q)(T(D))$.

4. EXPERIMENTS

Datasets and Query Workload: We have empirically studied the generality of widely used similarity search algorithms on two real world datasets: IMDb and DBLP. Since the computation of SimRank is extremely inefficient over large data graphs, we select a sample of of IMDb and DBLP datasets. The IMDb dataset consists of the movies produced in US from 2000-2012 with the rating of at least 6.0 (1038 nodes), their directors (846 nodes), top 100 actors who played in them, and their characters (2864 nodes). We construct a data graph of IMDb using the schema shown in Figure 1a and then transform it according to Figure 1b. The DBLP dataset consists of 22 database, data mining, and AI conferences during 2008-2012, their years, 100 most prolific authors of these conferences, and 2341 papers. We construct a graph for DBLP using the schema similar to Figure 2a, with *term* replaced by *year*, namely DBLP-1. We then transform DBLP-1 to a new graph, called DBLP-2, according to a schema shown in Figure 2b. Figure 3b illustrates this transformation over a fragment of DBLP data. We randomly select 30 actors for IMDb and 30 authors for the DBLP datasets as queries.

Similarity Methods and Parameters We compare the similarity ranking between the original data graph and a transformed data graph using Random Walk with Restart, SimRank, and PathSim. For RWR and SimRank, we use a decay factor $C = 0.8$. For PathSim, we use a meta-path AMA in IMDb and $ASMSA$ in Freebase where A, C, M, and S refer to *actor*, *character*, *movie*, and *starring*, respectively. We use $APCPA$ and ACA as a meta-path in DBLP-1 and DBLP-2 respectively where A is *author*, C is *conference* and P is *paper*.

Evaluation Metric: We adapt Kendall's tau metric to measure the difference between two ranked list. The metric penalizes a list if an answer occurs in a position that is different from its position in another list. Since each query may yield different sets of answers, we assume the missing elements from a list are ranked at the end of the other list. With this modification, each query can yield a different num-



(a) (b)

Figure 3: Transformation of a fragment of a DBLP. Labels A, C, Y and P are author, conference, year and paper, respectively.

ber of candidate answers. We normalized Kendall's tau by dividing it by its maximum value, $n(n-1)/2$, where n is the total number of unique elements in the two lists. If the two lists are identical, the measurement is 0. If one list is a reverse of the other list, then the measurement is 1. Furthermore, ranking methods generally returns too many answers but users usually focus more on top ranked answers. We perform the measurement over the top 10 and top 50 answers for each query.

Results:According to the formal definitions in section 3, the transformation for IMDb and Freebase are equivalent w.r.t. *actor*, and DBLP-1 and DBLP-2 are equivalent w.r.t. *author*. Table 1 shows the average ranking difference of top 10 and top 50 answers between equivalent representations of IMDb adn DBLP. None of the previously discussed methods are general under the transformations used in this experiment. RWR ranks the answers according to the degree distributions of nodes in the graph. All transformations in our experiment change the degree distribution, thus, the RWR computes a different score for each answer and return different rankings over equivalent databases. For example, RWR finds "Meryl Streep" to be more similar to the query actor "Nicolas Cage" than "Morgan Freeman" in IMDb, but it ranks "Morgan Freeman" higher than "Meryl Streep" in Freebase. In DBLP dataset, RWR finds "Christos Faloutsos" to be more similar to "Philip S. Yu" than "Jure Leskovec" in DBLP-1, but it finds the opposite in DBLP-2. SimRank usually prefers nodes with highly skewed degree distributions. Adding new nodes or new edges to the graph modify these values. Thus, the rankings returned by SimRank varies across original and transformed datasets. For example, SimRank ranks "Brian Cox" as the most similar actor to "Matt Demon" in IMDb but it ranks "George Clooney" the highest in Freebase. It ranks "Philip S. Yu" highest for query "W. Bruce Croft" in DBLP-1 but selects "Fabio Crestani" as the most similar author to "W. Bruce Croft" in DBLP-2.

	IMDb (10)	IMDb (50)	DBLP (10)	DBLP (50)
RWR	0.375	0.204	0.773	0.718
SimRank	0.418	0.264	0.626	0.673
PathSim	0.375	0.110	0.953	0.688

Table 1: Average ranking difference for top 10 and 50 answers over IMDb and DBLP.

PathSim is not general under the transformations on both IMDb and DBLP dataset. For example, PathSim finds "Maggie Gyllenhaal" to be the most similar actor to "Nicolas Cage" in Freebase, but returns "Michael Shannon" as the most similar actor to "Nicolas Cage" in IMDb. This difference is due to the fact that "Nicolas Cage" has played multiple roles in some films. IMDb graph contains only one path between a *movie* and each of its *actor*s regardless of the number of roles the actor has in that movie. However, in Freebase, the number of paths between a movie and each of its actors is more than one as the path between *actor* and *movie* passes through a *starring* node which is repeated once per each character that the actor played in the movie. For instance, if an actor has played two characters in a movie, there are two paths between that actor and the movie in Freebase. As PathSim uses the number of paths between two nodes to compute their similarities it return different results over these equivalent databases. Since DBLP-2 contains some duplicate nodes, the number of paths between some entities are larger in DBLP-2 than the number of paths between the same entities in DBLP-1. For instance, PathSim finds "Philip S. Yu" to be the most similar author to "Jiawei Han" in DBLP-1, but it finds "Charu C. Aggarwal" to be the most similar one to "Jiawei Han" in DBLP-2.

5. CONCLUSION AND FUTURE WORK

We introduced the problem of representation independent similarity search over graph data and proposed a formal framework to measure the degree of representation independence for a similarity search algorithm. We analyzed and empirically studied the representation independence of some widely used similarity search algorithms and show that although some of these methods may be more representation independent than others, none of them are representation independent. We plan to leverage our formal framework and findings in order to develop a more general similarity search algorithm over graph data.

6. ACKNOWLEDGMENTS

Yodsawalai Chodpathumwan is supported by NSF grants CCF-0938071, CCF-0938064, and CNS-0716532.

7. REFERENCES

[1] E. Codd. Does Your DBMS Run By the Rules? *ComputerWorld*, 1985.

[2] R. Hull. Relative Information Capacity of Simple Relational Database Schemata. In *PODS*, 1984.

[3] G. Jeh and J. Widom. SimRank: A Measure of Structural-context Similarity. In *KDD*, 2002.

[4] Y. Sun, J. Han, X. Yan, S. P. Yu, and T. Wu. PathSim: MetaPath-Based Top-K Similarity Search in Heterogeneous Information Networks. In *VLDB*, 2011.

[5] A. Termehchy et al. How Schema Independent Are Schema Free Query Interfaces? In *ICDE*, 2011.

[6] H. Tong and C. Faloutsos. Center-Piece Subgraphs: Problem Definition and Fast Solutions. In *KDD*, 2006.

[7] P. Zhao, J. Han, and Y. Sun. P-Rank: A Comprehensive Structural Similarity Measure over Information Networks. In *CIKM*, 2009.