# Universal-DB: Towards Representation Independent Graph Analytics

Yodsawalai Chodpathumwan, Amirhossein Aleyasen
University of Illinois
ychodpa2,aleyase2@illinois.edu

Arash Termehchy
Oregon State University
termehca@oregonstate.edu

Yizhou Sun
Northeastern University
yzsun@ccs.neu.edu

## ABSTRACT

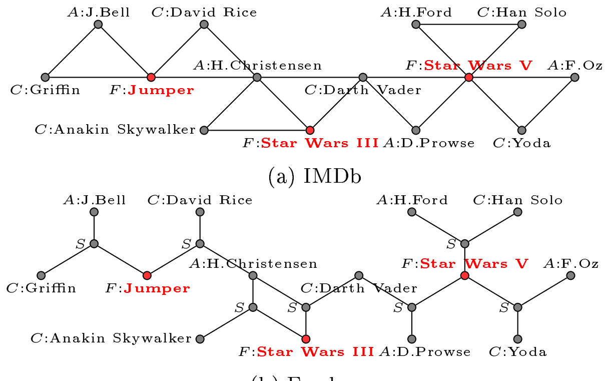
Graph analytics algorithms leverage quantifiable structural properties of the data to predict interesting concepts and relationships. The same information, however, can be represented using many different structures and the structural properties observed over particular representations do not necessarily hold for alternative structures. Because these algorithms tend to be highly effective over some choices of structure, such as that of the databases used to validate them, but not so effective with others, graph analytics has largely remained the province of experts who can find the desired forms for these algorithms. We argue that in order to make graph analytics usable, we should develop systems that are effective over a wide range of choices of structural organizations. We demonstrate *Universal-DB* an entity similarity and proximity search system that returns the same answers for a query over a wide range of choices to represent the input database.

## 1. INTRODUCTION

Finding similar or strongly related entities is a fundamental problem in graph data management and analytics [5, 7, 9]. Entity proximity and similarity algorithms are used as a building block for several important graph analytics tasks, such as community detection, link prediction, and subgraph matching [9]. Since the properties of *similar* or *related* entities are not precisely defined in the query, similarity and proximity search algorithms use intuitively appealing link-based heuristics to choose, from among all possible answers, those that are most similar or related to the query entity [5, 7, 9]. The power of these algorithms remains out of the reach of most users, however, as these tools are usable only by highly trained database analysts who can predict which algorithms are likely to be effective for particular representations of the data, or who are able to customize these algorithms to satisfy their information needs over a new database. To see why, consider the following example.

*Proceedings of the VLDB Endowment,* Vol. 8, No. 12
Copyright 2015 VLDB Endowment 2150-8097/15/08.

EXAMPLE 1.1. *Figure 1a and 1b show excerpts of IMDb (imdb.com) and Freebase (freebase.com) about the same set of movies, their characters, and the actors who played them, respectively. IMDb and Freebase use different representations to express the same relationships between movies, characters, and actors. Random Walk with Restart (RWR) [9] and SimRank [5] are two well-known linked-based similarity and proximity search algorithms. RWR evaluates how likely an entity will be visited if a random surfer starts and keeps re-starting from the query entity. SimRank evaluates the similarity between two entities according to how likely two random surfers will meet each other if they start from the two entities. RWR and SimRank find Star Wars III more similar to Star Wars V than to Jumper in Figure 1a, but find Star Wars III to be more similar to Jumper than to Star Wars V in Figure 1b.*



(a) IMDb

(b) Freebase

Figure 1: Fragments of IMDb and Freebase, where $A$, $C$, $F$, and $S$ refer to *actor*, *character*, *film* and *starring*, respectively.

Thus, users generally have to *restructure* or *wrangle* their databases to some *proper representations*, to effectively use similarity and proximity search algorithms,i.e., deliver the insights that a domain expert would judge as relevant. To make matters worse, these algorithms do not normally offer any clear description of their desired representations and database analysts have to rely on their own expertise and/or do trial and error to find such representations. Nevertheless, we want our database analytics algorithms to be used by ordinary users, not just experts. Further, the structure of large-scale databases constantly evolve, and we want to move away from the need for constant expert attention to keep exploration algorithms effective.

One approach to achieve representation independence is to define a *universal representation* to which all possible representations of a database can be transformed and develop algorithms that are effective over this representation. Nevertheless, the experience gained from the idea of universal relation indicates that such representation may not always exist [1]. Further, it is not clear if one can successfully enforce data curators to represent their data in a certain way.

Another method is to run certain algorithm over all representations of a database and select the representation(s) with the most accurate answers. Nonetheless, it is undecidable to compute all representations of a database [3]. If one restricts her search to a particular type of representations, the database may have still a large number of representations. For example, a relational table may have exponential number of vertical decompositions. This number may be larger for the graph databases that do not follow a fixed schema. It may take a great deal of time and resources to convert a large and/or evolving database to various representations. Further, this method requires training data, which is not generally available in many database analytics tasks, such as entity similarity search.

To cope with organizational heterogeneity and evolution in large-scale data, it is time to move beyond database analytics algorithms that are effective only over certain representations of the database. We demonstrate *Universal-DB*; an entity similarity and proximity search system that delivers the same answers for a query over various representation of a graph database. In particular,

- We illustrate different types of organizational heterogeneity that are observed in real-world databases. We demonstrate how well-known entity similarity search algorithms deliver different results for the same query over these representational choices.
- We show that Universal-DB returns the same answers for a query over a wide range of choices for representing the data and explain its underlying techniques.

## 2. RELATED WORK

The architects of relational model have argued for logical data independence, which oversimplifying a bit, means that an exact query should return the same answers no matter which logical schema is chosen for the data [1]. The idea of representation independence extends the principle of logical data independence for database analytics algorithms. Researchers acknowledge the growing need for users to transform their data and provide systems to help them with wrangling their data [6]. We address the same problem but using a difference approach: *eliminate the need to wrangle the data.* Researchers have developed effective keyword search techniques that are robust across multiple tree-shaped XML representations of the same information [8]. We built on this line of work by developing representation independent systems over graph data models and entity similar and proximity search problem.

## 3. REPRESENTATION INDEPENDENCE

Let domain *Dom* be a countably infinite set of values, e.g. strings or integers. Also, let $L$ be a finite set of (string) labels, e.g. *actor* and *film*. A database $D$ is a graph $D = (V, E, \mathcal{L}, \mathcal{A})$, where $V$ is the set of nodes in the graph, $E \subseteq V \times V$ is the set of edges in the graph, $\mathcal{L}$ is a total function from $V$ to $L$, and $\mathcal{A}$ is a function from $V$ to *Dom*. Figure 1 show fragments of some databases. The function $\mathcal{L}$ assigns a label to every node. The label of a node shows the *semantic type* of the information that the node represents. Each node in the database represents an entity, e.g. $film$:Star Wars III in Figure 1b, or a relationship between entities, e.g. *starring* in Figure 1b.

*Database transformations* (transformation for short) have been used to compare the information content of different databases [3]. Transformation $\tau$ over database $D$ is a (computable) function that maps $D$ to another database $\tau(D)$. If $\tau$ is *invertible*, we can reconstruct the information available in $D$ given $\tau(D)$. For example, the transformation between IMDb and Freebase DBs in Figure 1 is invertible, as it replaces every *triangular* subgraph whose nodes represent entities of types *film*, *actor*, and *character* in a graph database with a *star* subgraph where these entities are connected via a single instance of type *starring*.

Furthermore, we should make sure that users can pose the same set of queries over databases $D$ and $\tau(D)$. Similarity search queries over a database $D$ are entities of $D$. We say that $\tau$ is *query preserving* iff $D$ and $\tau(D)$ essentially contain the same set of entities. For example the transformation between IMDb and Freebase DBs in Figure 1 is query preserving because it does not introduce any new entity or remove any existing entity in the input databases.

If a transformation is both invertible and query preserving, it is *similarity preserving*. A similarity preserving transformation maps a databases $D$ to a database $\tau(D)$ that has the same information and the same set of possible queries. It possible to design an effective similarity search algorithm that returns essentially the same answers for every query over $D$ and $\tau(D)$. Given an entity, a similarity search algorithm returns a ranked list of entities that are most similar to the query entity. Intuitively, a representation independent similarity search algorithm should return the same list of entities to the same query across databases that represent the same information. More formally, algorithm $A$ is *representation independent* under bijective transformations $\tau$ iff it returns the same ranked list of entities over databases $D$ and $\tau(D)$ for the same query. Algorithm $A$ is representation independent under a set of query preserving transformations $T$ iff it is representation independent under all transformations in $T$.

## 4. UNIVERSAL-DB

### 4.1 Representational Shifts

We consider the following two transformations that frequently occur in real-world databases.

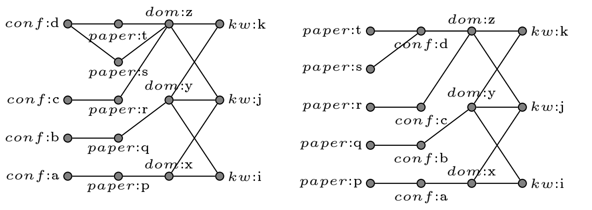
**Relationship Reorganizing Transformations:** Real-world graph databases may use either edges or nodes to represent the same relationships. For example, the relationship between entities actor, character, and film is represented by connecting them using some edges in Figure 1a. But the same relationship between these entities is depicted by connecting them to a node of label *Starring* in Figure 1b. One may also use nodes without values to categorize related nodes, which help users understand the structure of the database more easily. For instance, in a movie data set from the Niagra project[1], all actors of a movie are grouped under a node of label *cast*. Intuitively, this node does not add any new information to the database as the actors that play in a movie are already connected to the movie using some edges. But it helps users to navigate and explore the

[1]research.cs.wisc.edu/niagara/data.html

database easier. Using nodes without values to represent relationships between entities or categorize them is prevalent in real-world graph databases [4].

**Entity Rearranging Transformations:** Given some constraints in the database, people may connect same entities in different orders. For example, consider a bibliographical database of Microsoft Academic Search[2] (MAS) and its alternative representation as shown in Figure 2. Both databases contain entities of semantic types paper, conference, domain, and keyword. The domains of papers and conferences show their areas, e.g., *database* and *information retrieval.* The keyword entities contain the keywords of domains, e.g., *indexing* and *query processing* for *database* domain. Each paper is published in only one conference and each conference belongs to only one domain. In original MAS, the database connects each paper to its conference and its main domain. However, the alternative representation connects each paper to its conference and each conference to its corresponding domain instead. Both representations connect each domain to all related keywords.

Relationship reorganizing and entity rearranging transformations are similarity preserving [2].



(a) Original representation (b) Alternative representation

Figure 2: Fragments of some representations for MAS data

## 4.2 Robustness of Current Algorithms

To the best of our knowledge, the most frequently used methods for similarity search on graph database are based on Random Walk, e.g., RWR [9], Pairwised Random Walk, e.g., SimRank [5], or relationship-constrained framework, e.g., PathSim [7]). There are other similarity measures, such as common neighbors, $Katz_\beta$ measure, hitting time, and commute time, which can be considered as special cases of aforementioned heuristics. Methods that use random walk and pairwise random walk leverage the topology of a graph database to measure the degree of similarities between entities. Relationship reorganizing and entity rearranging transformations may remove many of edges in a database and add a large number of new nodes and edges to the database. Thus, they may radically modify the topology of the database. For example, relationship reorganizing transformations may drastically change the degree of a node and modify the probability that random surfers visit the node. Hence, these methods cannot always return the same answers over the original and the transformed database for the same query. Our example in section 1 illustrates that RWR and SimRank are not representation independent under relationship reorganizing transformations.

PathSim measures the similarity between entities over a *given relationship* [7]. For example, it may compute the similarity of two movies in a movie database based on the actors that have played in those movies. It defines relationships between entities as a sequence of labels, i.e., meta-paths. For example, the relationship between two movies in Figure 1b according to their common actors is represented by meta-path ($film$, $starring$, $actor$, $starring$, $film$) PathSim computes the score using the number of paths between entities that follow the given meta-path.

However, the relationship reorganization and entity rearranging transformations can change the number of the instances for a meta-path. Thus, PathSim may not find the same answers over the database and its transformation. For example, assume that a user likes to find conferences similar to *conf*:b based on their common keywords in the database fragments in Figures 2a and 2b. She needs to use meta-path ($conf$, $paper$, $dom$, $kw$, $dom$, $paper$, $conf$) and meta-path ($conf$, $dom$, $kw$, $dom$, $conf$) in Figure 2a and 2b, respectively. The meta-path ($conference$, $domain$, $conference$) has only one instance for each pairs of conference entities but the number of instances of meta-path ($conf$, $dom$, $kw$, $dom$, $conf$) between a pair of conferences varies by the number of papers published in the related conferences. Hence, PathSim finds *conf*:a and *conf*:c to be equally similar to *conf*:b in Figure 2b, but it finds *conf*:a to be more similar to *conf*:b than *conf*:c in Figure 2a.

## 4.3 Robust Proximity and Similarity Algorithm

Universal-DB uses a robust relationship-constrained similarity and proximity search algorithm. We have observed that a relationship that is expressed as a simple path in one representation of a database may be represented using forms other than simple paths in its transformations. Hence, Universal-DB considers a richer set of expressions for relationships in databases. Representational modifications may convert a simple path to a path with repeating nodes, i.e., non-simple path. For example, the relationship ($conf$, $paper$, $dom$, $kw$, $dom$, $paper$, $conf$) in Figure 2a is represented by ($conf$, $paper$, $conf$, $dom$, $kw$, $dom$, $conf$,$paper$, $conf$) in Figure 2b. Hence, Universal-DB considers the relationships represented by both simple and non-simple paths in the database. Further, some relationships in the original database cannot be represented as neither simple nor non-simple paths in its transformations. For example, there is not any path in Figure 2a that represent the relationship expressed through $p$ = ($conference$, $domain$, $conference$) in the database fragment of Figure 2b. The relationship that follow ($conference$, $paper$, $domain$, $paper$, $conference$) contains a bit more information than $p$ as it contains the set of papers published in each conference. Hence, Universal-DB considers the relationships in the database that are represented by paths that may jump over some nodes in the database. For instance, it considers the relationships represented by ($conference$, $\overline{paper}$, $domain$, $\overline{paper}$, $conference$) in Figure 2b, where $\overline{paper}$ indicate that conference and domain are connected through some entities of type paper without any regard to the strength of such connection, i.e., the number of papers published in the conference. These relationships contain exactly the same information as $p$.

Representational shifts may introduce or remove some spurious relationship instances to or from a database. Figure 3 shows excerpts of SNAP[3] and DBLP databases, where the citation relationships between papers are expressed by

[2]academic.research.microsoft.com

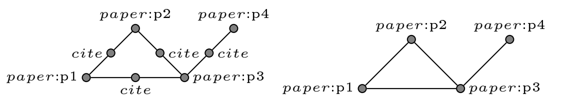
[3]snap.stanford.edu/data

| | RWR | SimRank | PathSim | UnivDB |
|---|---|---|---|---|
| Top 5 | .134 | .578 | .327 | 0 |
| Top 10 | .141 | .493 | .296 | 0 |

Table 1: Average ranking difference for DBLP to SNAP transformation

edges in SNAP and through nodes without any value with label *cite* in DBLP. Assume that a use likes to find strongly related papers by leveraging the fact that these papers may frequently cite some common papers. There is only one instance of the relationship (*paper*, *paper*, *paper*) between *paper*:p1 and *paper*:p2 in Figure 3b. The same relationship is represented by (*paper*, *cite*, *paper*, *cite*, *paper*) Figure 3a, which has three instances. The additional instances are (*paper*:p1, *cite*, *paper*:p1, *cite*, *paper*:p2) and (*paper*:p1, *cite*, *paper*:p2, *cite*, *paper*:p2). These instances do not provide any information in addition to the one that the other instance of this relationship already provides. Universal-DB recognizes spurious instance by noting that in these instances at least one entity and its closest entity are equal and ignores them.

Since users may not be familiar with the structure of the database, Universal-DB aggregates the similarity scores between each pairs of entities over all relationships between these entities so users do not need to supply the template of the relationships. Because Universal-DB considers relationships expressed by structures beyond simple paths, it may take a long time to compute a similarity score between two entities. We have shown that it is sufficient to consider a minimal subset of all such paths in a database to have a robust similarity search algorithm, and thus, compute the scores efficiently [2]. Universal-DB is provably representation independent under relationship reorganizing and entity rearranging transformations [2]. Tables 1 and 2 show the average ranking differences for answers of RWR, SimRank, PathSim, and Universal-DB based on transformations in Figures 3 and 4 using normalized Kendall's Tau [8] over workloads of 100 queries, respectively. The score of 0 indicates that there is no ranking difference between the two lists from source and transformed databases, and the score of 1 indicates that the two ranked lists are completely opposite. Universal-DB returns as effective or more effective results than other similarity and proximity search algorithms [2].



(a) DBLP (b) SNAP

Figure 3: DBLP and SNAP Citation graphs.

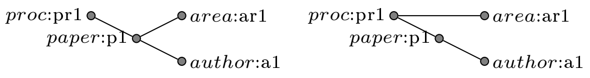


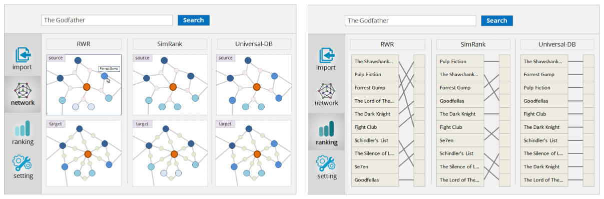
Figure 4: Two representations for a bibliographic database.

## 5. DEMONSTRATION

In our demonstration, we will present examples of real-world databases, such as IMDb, Freebase, DBLP, and SNAP that provide different ways of representing the same information. Our demonstration helps the audience to compare representation independence of well-known similarity search algorithms: RWR, SimRank, PathSim over the types of representational shift discussed in Section 4.1. In addition, they will try and examine the robustness of Universal-DB over several databases. Users will select a database, an entity similarity search algorithm, and submit a query to the database. Our prototype computes the results for the query the selected similarity search algorithm. Our user interface provides two views. In the first view, *network view*, shown in Figure 5a, the user can see the graph visualization of the query entity (shown in orange) as well as the answers (shown in blue) for the algorithm over different graph representations of the database. The nodes with higher ranks in the results have higher color contrast in the network view. This view helps the user to gain some insight on how the structures of a representation affect the answers of the query. Users may also run queries using Universal-DB and observe how it returns the same answers over different representations of a database. Users may also compare the rankings delivered by different similarity search methods over different representations of the same dataset using the *ranking view*, which is shown in Figure 5b. The same entities in different ranked lists are connected using some lines. Algorithms that are more robust on a type of transformation have fewer number of crossing between the connection lines.

| | RWR | SimRank | PathSim | UnivDB |
|---|---|---|---|---|
| Top 5 | .475 | .401 | .632 | 0 |
| Top 10 | .415 | .345 | .605 | 0 |

Table 2: Average ranking difference for different representations of the bibliographic database



| RWR | SimRank | Universal-DB |
|---|---|---|
| The Shawshank... | Pulp Fiction | The Shawshank... |
| Pulp Fiction | The Shawshank... | Forrest Gump |
| Forrest Gump | Forrest Gump | Pulp Fiction |
| The Lord of The... | Goodfellas | Goodfellas |
| The Dark Knight | The Dark Knight | Fight Club |
| Fight Club | Fight Club | The Dark Knight |
| Schindler's List | Se7en | Schindler's List |
| The Silence of L... | Schindler's List | The Silence of L... |
| Se7en | The Silence of L... | The Dark Knight |
| Goodfellas | The Lord of The... | The Lord of The... |

(a) Network View (b) Ranking View

Figure 5: Universal-DB screenshots

## 6. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1994.

[2] Y. Chodpathumwan et al. Representation Independence of Structural Proximity and Similarity Algorithms. Technical report, Oregon State University, 2015.

[3] W. Fan and P. Bohannon. Information Preserving XML Schema Embedding. *TODS*, 33(1):1–44, 2008.

[4] A. Hogana, M. Arenas, A. Mallea, and A. Polleres. Everything you always wanted to know about blank nodes. *Web Semantics*, pages 42–69, 2014.

[5] G. Jeh and J. Widom. SimRank: A Measure of Structural-context Similarity. In *KDD*, pages 538–543, 2002.

[6] S. Kandel et al. Wrangler: Interactive visual specification of data transformation scripts. In *CHI*, 2011.

[7] Y. Sun et al. PathSim: MetaPath-Based Top-K Similarity Search. In *VLDB*, pages 992–1003, 2011.

[8] A. Termehchy et al. Design Independent Query Interfaces. *TKDE*, 24(10):1819–1832, 2012.

[9] H. Tong et al. Fast Random Walk with Restart and its Applications. In *ICDM*, pages 613–622, 2006.