SimFusion: Measuring Similarity using Unified Relationship Matrix

Wensi Xi, Edward A. Fox, Weiguo Fan
Virginia Tech
Blacksburg, VA, 24061
{xwensi, fox, wfan}@vt.edu

Benyu Zhang, Zheng Chen
Microsoft Research Asia
Beijing, China, 100080
{byzhang, zhengc}@microsoft.com

Jun Yan[1], Dong Zhuang[2]
[1]Beijing University
[2]Beijing Institute of Technology
Beijing, China, 10080
[1]yanjun@math.pku.edu.cn,
[2]zhuangdong@bit.edu.cn

ABSTRACT
In this paper we use a Unified Relationship Matrix (***URM***) to represent a set of heterogeneous data objects (e.g., web pages, queries) and their interrelationships (e.g., hyperlinks, user click-through sequences). We claim that iterative computations over the ***URM*** can help overcome the data sparseness problem and detect latent relationships among heterogeneous data objects, thus, can improve the quality of information applications that require combination of information from heterogeneous sources. To support our claim, we present a unified similarity-calculating algorithm, ***SimFusion***. By iteratively computing over the ***URM***, ***SimFusion*** can effectively integrate relationships from heterogeneous sources when measuring the similarity of two data objects. Experiments based on a web search engine query log and a web page collection demonstrate that ***SimFusion*** can improve similarity measurement of web objects over both traditional content based algorithms and the cutting edge ***SimRank*** algorithm.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information search and retrieval; G.2.2 [**Discrete Mathematics**]: Graph Theory.

General Terms
Algorithms, Experimentation

Keywords
SimFusion, Information retrieval, Information integration.

1. PROBLEM DEFINATION
Concerned with the information explosion that was gathering momentum after WWII, Vannevar Bush gave us a vision of linked information in 1945, which has helped inspire the *WWW* [4]. Today, the success of the *WWW*, and many other information technologies, leaves us with a much larger information explosion. To survive this challenge, we must learn to manage that information more effectively, and so must have new, more powerful models and techniques for information integration, in complex contexts.

Authors, documents, users, metadata, and other types of entities present in scientific/scholarly domains; as well as *web pages, users, queries,* and other entities found in web domains; all can be considered as data objects containing information. The information may characterize content features of individual objects, as well as relationships between objects, from the same or different types of sources. In the web domain, for example, we know that users browse web pages and issue queries. Queries, in turn, may lead to the referencing of web pages. These three operations (*browsing*, *issuing*, and *leading* to a page reference) are relationships that connect different types of objects. We also know that users are connected by their social relationships, web pages are connected by hyperlinks, and queries are connected by their content similarities. These three latter connections can be viewed as relationships within the same type of objects.

Modern information applications, such as searching and document clustering, mainly use three approaches to represent information objects and the interrelationships involved:

1. Spaces: Vector and probability spaces implicitly use a set of features from objects, e.g., to locate them in an n-dimensional space [30].
2. Databases: Relational databases operate on objects and their relationships (represented using sets of attributes) [5] [12].
3. Networks: Belief, inference, and spreading activation networks (e.g., with semantic networks) use nodes and arcs to represent objects, attribute values, and relationships [24] [28].

However, most information applications being used today take only one of the three approaches to analyze one kind of relationship within the same type of objects (e.g., document clustering, web link analysis) or only between two types of objects (e.g., web search, collaborative filtering). These applications are hard to change. We run into problems when the users of these applications require more accurate models of reality, wherein the number of types and sub-types of objects that must be handled expand rapidly (e.g., considering both queries and users when clustering web pages), and the relationships between different types of objects (e.g., considering both reference and browsing relationships when analyzing behaviors associated with web pages) grow tremendously. More specifically, the problem we are facing can be stated as:

"How can the broad variety of heterogeneous data and relationships be effectively and efficiently combined to improve the performance of various information applications?"

2. OBJECTIVES AND HYPOTHESIS
The purpose of our research is to find effective and efficient means of combining relationships from multiple heterogeneous data sources, thus overcoming limitations of the three traditional approaches discussed above.

In this paper, we claim that the *Unified Relationship Matrix (URM)* (described in detail in Section 4.2) can be used to represent relationships from multiple and heterogeneous information sources. We further claim that iterative computation, over the ***URM***, will improve the quality and utility of information from heterogeneous sources for a variety of search related information applications. To prove this, we have focused our research on a specific information application: similarity calculation involving heterogeneous data objects.

The underlying hypothesis is that: Relationships can be represented through matrices accurately, with either binary or real-valued weights. Matrices representations of different types of relationships are sometimes complementary. A matrix representation of a single relationship may be sparse, but when reinforced by other types of relationships represented in complementary matrices, the information it contains may be more dense and helpful. We contend that matrix representation and matrix processing are effective approaches for combining relationships from difference sources. Figure 1 gives a simplified illustration of our hypothesis. Note that as a result of our methods, the bottom row of matrices, which can be used for various applications, is presumably of higher quality (e.g., less sparse, due to the addition of accurate new values, and so more effective).

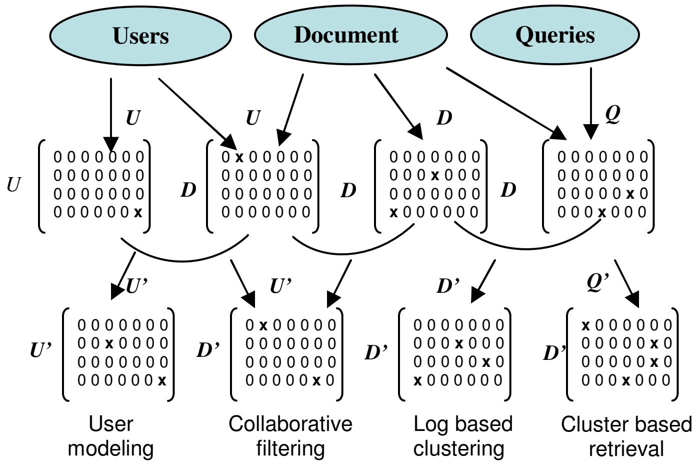

Figure 1: Matrix representations of relationship integration

3. RELATED WORKS

Many works have focused on using a single type of relationship when calculating the similarity of data objects. Approaches that calculate the similarity of document-query objects using document-term relationships include: Vector Space Model (VSM) [26], Generalized Vector Space Model [30], Latent Semantic Indexing [11], query expansion [25], and dynamic vector space modification [2]. These can be viewed as variations of a general algorithm that projects documents and queries into a vector space using singular vectors; they differ in how the vectors are constructed and how weights are assigned.

Reference relationships also can help us gauge the similarity among scientific papers, e.g., by considering co-citations [27] and bibliographic coupling [17]. These in turn have been used to cluster scientific journals [19]. In the *WWW*, relationships among pages (e.g., based on hyperlinks) were used to calculate the similarity of web objects [10], and helped in web clustering tasks [1] [29]. Relationships between people and artifacts have been used to calculate the preferences of people for artifacts, in collaborative filtering [14] and recommender systems [23].

Most works discussed above only consider a single type of relationship when calculating the similarity of data objects. Some recent works [7] [15] proposed using multiple relationships to calculate the similarity between data objects. However, these works differ from ours mostly in 1) how different kinds of attributes (relationships) are combined, 2) the representation of relationships, and 3) theoretical foundations. Most recently, [20] and [32] focused on taking advantage of mutual reinforcement effects, and iteratively calculate the similarity of data objects. However, these works lack a theoretical foundation, and their integration methodology also is different from ours. Although the representation and calculation in [8] and [31] are similar to our work, they differ in how similarity is calculated using matrices (see Section 5). In [8], the expensive iterative computations limit its online use. [31] differs in that its purpose is to find the attribute values of single data objects, not the similarity of data objects. We go beyond earlier works by providing a unified representation of different types of relationships, by developing a framework and a solid theoretic foundation for calculating with regard to relationships between data objects, and by considering multiple types of relationships as well.

4. DEFINITIONS AND EXAMPLES

4.1 Terminology

We first give simple definitions for key terms as they will be used in the rest of this paper:

Data Type: A **data type** refers to a class of objects, defined by a set of characteristic features (e.g., *user* has a set of features including name, gender, etc.). A **data object** is an instance of a data type.

Data Space: A **data space** is a set of **data objects** with the same data type (e.g., *web pages* in the Internet). Table 1 gives examples of related data spaces and data objects.

Homogeneous/Heterogeneous: Each data space is **homogeneous** within itself, but **heterogeneous** with respect to other data spaces.

Intra-type relationship: connects information objects within a homogeneous data space (e.g., hyperlinks within web pages).

Inter-type relationship: connects information objects across heterogeneous data spaces (e.g., users *issue* queries and *browse* web pages. The *issuing* and *browsing* activities can be regarded as inter-type relationships connecting the user and web page data spaces, and the user and query data spaces, respectively.).

Table 1: Some relationships in information applications

Data Spaces	Examples of Data Objects
People	Authors, editors, users
Terms	Concepts, stems, words
Queries	Natural language queries, Boolean queries
Documents	Journals papers, refereed conference papers

4.2 Unified Relationship Matrix *(URM)*

4.2.1 Definition

The formal definition of the ***Unified Relationship Matrix*** (***URM***) that represents both inter- and intra-type relationships among heterogeneous data objects in a unified manner is given below.

Suppose there are t different data spaces S_1, S_2… S_t. Data objects within the same data space are connected via intra-type relationships $R_i \subseteq S_i \times S_i$. Data objects from two different data spaces are connected via inter-type relationships $R_{ij} \subseteq S_i \times S_j$ $(i \neq j)$. The intra-type relationships R_i can be represented as an $m \times m$ adjacency matrix L_i (m is the total number of objects in data space S_i). Inside matrix L_i cell l_{xy} represents the inter-type relationship from the x_{th} object to the y_{th} object in the data space S_i. The inter-type relationship R_{ij} can be represented as an $m \times n$ adjacency matrix L_{ij} (m is the total number of objects in S_i, and n is the total number of objects in S_j), where the value of cell l_{xy} represents the inter-type relationship from the x_{th} object in S_i to the j_{th} object in S_j. If we merge N data spaces into a unified data space U, then previous inter- and intra-type relationships are all part of intra-type relationships R_u in data space U. Suppose L_u is the adjacency matrix of R_u, then L_u is a square matrix. We define the Unified

Relationship Matrix L_{urm} as a matrix that combines all the relationship matrices, as given in Eq. (1):

$$L_{urm} = \begin{vmatrix} L_1 & L_{12} & \cdots & L_{1N} \\ L_{21} & L_2 & \cdots & L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & L_N \end{vmatrix} \qquad (1)$$

4.2.2 Real World Examples

The ***URM*** provides a generalized way of viewing data objects and their relationships. In the ***URM***, different types of objects are treated as elements of a "unified" data space. Previous inter- and intra-type relationships are each considered as a generic intra-type relationship that connects data objects in the "unified" data space.

The ***URM*** can be used to explain a variety of real world information application scenarios. For example, if we only consider one data space, of web pages, and one type of intra-type relationship, the hyperlink relationship, then the ***URM*** is reduced to the link adjacency matrix of the web graph, upon which advanced web link analysis works such as [3] and [18] are based.

Consider another example. If we have two data spaces, documents and terms, then an inter-type relationship is defined when a document contains a term or a term is contained by a document. A ***URM*** can be built as in Eq. (2).

$$L_{urm} = \begin{vmatrix} 0 & L_{dt} \\ L^T{}_{dt} & 0 \end{vmatrix} \qquad (2)$$

L_{dt} is the traditional document-term matrix used in the VSM [26]. The 0 sub-matrices on the diagonal indicate that we have no prior knowledge of intra-type relationships within the document or term space. All the information applications that manipulate the document-term matrix can still be used on L_{urm}. Furthermore, the intra-type relationship of the document and term spaces can be obtained by simply multiplying L_{urm} with itself: $L'_{urm} = L_{urm} \times L_{urm} = \begin{vmatrix} L_d & 0 \\ 0 & L_t \end{vmatrix}$, where L_d and L_t correspond to the document pair-wise similarity matrix and term pair-wise similarity matrix obtained by most traditional VSM similarity calculations. By adding L'_{urm} and L_{urm}, we can have a complete ***URM*** for the document and term spaces: $\begin{vmatrix} L_d & L_{dt} \\ L^t_{dt} & L_t \end{vmatrix}$. This specific matrix that combines document pair-wise and term pair-wise relationships with traditional document-term relationships was named by Davidson [8] as the "generic augmented matrix".

5. ALGORITHM

5.1 Similarity Reinforcement Assumption

Our basic assumption is that: "the similarity between two data objects can be reinforced by the similarity of related data objects from the same and different spaces", as is illustrated below:

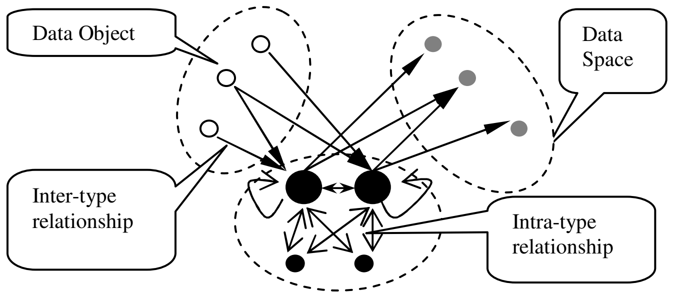

Figure 2: Illustration of similarity reinforcement assumption

As can be seen in Figure 2, the similarity between two data objects (big black nodes in the center) is reinforced by relationships from the same type of related data objects (small black nodes) as well as the relationships (both inbound and outbound) from different types of data objects (white and gray nodes).

Suppose there are N different data spaces $X_1, X_2, \ldots X_N$. Data objects in the same space are related via intra-type relationships $R_i \subseteq X_i \times X_i$. Data objects from different spaces are related via inter-type relationships $R_{ij} \subseteq X_i \times X_j$ $(i \neq j)$. The relationships being considered are similar in nature. $S_{ij}(x,y)$ represents the similarity between object x from space i and object y from space j. $R_{ij}(x,y)$ represents the inter- $(i=j)$ or intra-type $(i \neq j)$ relationship from object x in space i to object y in space j, while a and b are any data objects in any data spaces under the condition that x is related to a and y is related to b. Then the similarity reinforcement assumption can be mathematically presented as:

$$S_{ij}^{new}(x,y) = \alpha S_{ij}^{original}(x,y) + \beta \sum_{(\forall k, and \forall a \in k),(\forall l, and \forall b \in l)} R_{ik}(x,a) R_{jl}(y,b) S_{kl}^{original}(a,b) \qquad (3)$$

where α and β are positive parameters used to adjust (during the reinforcement process) the relative importance of the original similarity of objects x and y with the importance of the similarity reinforced by inter- and intra-type relationships. If we use a set of positive parameters λ_{ij} to represent the relative importance of similarity reinforced from data space i to data space j, and consider the amount of original similarity value involved in this process as the similarity value reinforced via a special intra-type relationship that leads to the data object itself (indicated in Figure 2), the similarity reinforcement assumption can be represented as:

$$S_{ij}^{new}(x,y) = \lambda_{ii} R_{ii}(x,x) \lambda_{jj} R_{jj}(y,y) S_{ij}^{original}(x,y) + \sum_{\forall a \in \forall k, \forall b \in \forall l} \lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}^{original}(a,b) \qquad (4)$$

Eq. (4) can be reduced to Eq. (5):

$$S_{ij}^{new}(x,y) = \sum_{\forall a, \forall b} \lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}^{original}(a,b) \qquad (5)$$

Considering one data object's related objects in other data spaces as its mappings in those data spaces, the reason the similarity reinforcement process can yield better estimates is that the similarity of two data objects is measured in multiple perspectives (data spaces) instead of a single perspective. However, a pre-condition is that relationships involved are accurate and additive. Thus, care should be taken to avoid involving contradictory or ambiguous types of relationships.

5.2 The SimFusion Algorithm

Based on the "similarity reinforcement assumption", we develop a unified similarity calculation algorithm over a set of heterogeneous data spaces: "***SimFusion***". The name indicates that the similarity of two data objects is calculated using evidence from multiple sources (data spaces). It is formally described as:

Suppose there are N different spaces being considered, and a ***URM*** is developed in a similar way to Eq. (2) to represent the inter- and intra-type relationships as shown in Eq. (6):

$$L_{urm} = \begin{vmatrix} \lambda_{11} L_1 & \lambda_{12} L_{12} & \cdots & \lambda_{1N} L_{1N} \\ \lambda_{21} L_{21} & \lambda_{22} L_2 & \cdots & \lambda_{2N} L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N1} L_{N1} & \lambda_{N2} L_{N2} & \cdots & \lambda_{NN} L_N \end{vmatrix} \qquad (6)$$

Here L_i is the intra-type relationship matrix of data space i and L_{ij} is the inter-type relationship matrix from data space i to data space *j*. The sum of each row from any of the sub-matrices is normalized to 1. In cases that data object *x* from space *i* has no relationship to any data objects in data space *j* (all the elements in the i_{th} row of the matrix L_{ij} are zero), then each element in the i_{th} row of relationship matrix L_{ij} is set to *1/n*, where *n* is the $_{\text{total}}$ number of elements in space *j*. This is equivalent to using a random relationship to represent no-relationship. We also define a set of parameters λs to adjust the relative importance of different inter- and intra-type relationships, so that for any *i*, $\sum_{\forall j}\lambda_{ij}=1 \quad and \ \forall i,j \ \ \lambda_{ij}>0$. Thus, Eq. (6) is a row-stochastic matrix and can be rendered as a single step probability transformation matrix in a Markov Chain [16].

We also define a Unified Similarity Matrix (***USM***), S_{usm}, to represent the similarity values of any data object pairs from same or different data spaces at the beginning of the algorithm:

$$S_{usm}=\begin{vmatrix} 1 & s_{12} & \cdots & s_{1T} \\ s_{21} & 1 & \cdots & s_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1T} & s_{2T} & \cdots & 1 \end{vmatrix} \qquad (7)$$

Each element $s_{a,b}$ in S_{usm} represents the similarity value between data objects *a* and *b* in the unified space. *T* is the total number of objects in the unified space. Since each data object is always maximally similar to itself, we have $s_{ab}=1$ if $a=b$, and $0\leq s_{ab}\leq 1$, if $a\neq b$. S_{usm} is a symmetric matrix since $s_{ab}= s_{ba}$. We also define that the order of data objects presented in S_{usm} and L_{urm} are similar, that is, if the element l_{ab} in L_{urm} represents the relationship from object *a* to object *b*, then element s_{ab} in S_{usm} represents the similarity value between objects *a* and *b*. Having ***URM*** and ***USM*** defined, the similarity reinforcement assumption can be represented as:

$$S_{usm}^{new} = L_{urm}\, S_{usm}^{original}\, L_{urm}^{T} \qquad (8)$$

Eq. (8) is the basic similarity reinforcement calculation in the ***SimFusion*** algorithm. Eq. (8) can be continued in an iterative manner until the calculation converges or a satisfactory result is obtained, as shown in Eq. (9).

$$S_{usm}^{n} = L_{urm}\, S_{usm}^{n-1} L_{urm}^{T} = L_{urm}^{n}\, S_{usm}^{0}\, (L_{urm}^{T})^{n} \qquad (9)$$

The proof of convergence for Eq. (9) can be found in the appendix of [33]. The proof is based on the convergence proof of the PageRank [3] algorithm, by converting Eq. (9) into a power calculation of a matrix with a vector. It is worthwhile to note that, similar to the PageRank algorithm, the values in the ***USM*** after convergence are independent to the initial values in the ***USM*** at the beginning of the calculation. In practice, the initial ***USM*** is often set to be an identity matrix. It also is important to note that the similarity of a data object to itself (i.e., the values in the diagonal positions of S^{n}_{usm}) derived during the iterative calculation in Eq. (9) may not be equal to 1 and even may be smaller than the similarity between two data objects (i.e., values in non-diagonal positions in S^{n}_{usm}). We argue that the S_{usm} derived during the iterative calculation can be interpreted more precisely as the reliability of the similarity evidence rather than as exact similarity values. For example, if a data object (or two data objects) is related to a set of less similar data objects, then the similarity of the data object (or the two data objects) is less reliable when used as evidence to reinforce the similarity of data objects related to it in the next iteration than if the data object (or two data objects) is related to a set of very similar data objects. Thus, in S^{n}_{usm} the similarity of a data object to itself may not be equal to 1 and may be even smaller than the similarity of some object pairs, as is illustrated in Figure 3:

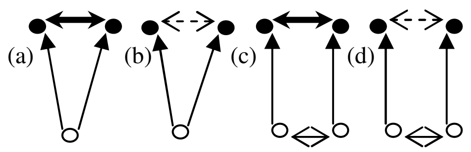

Figure 3: Illustration of similarity confidence

In Figure 3, the white nodes at the bottom represent the objects being considered. The black nodes on the top represent the objects that the white objects relate to. Arrows between two black nodes or two white nodes indicate their degree of similarity through their thickness. Thus, (a) is a high similarity-confidence object, (b) is a weak similarity-confidence object, (c) is a high similarity-confidence object pair, and (d) is a low similarity-confidence pair.

The theoretical foundation, and the time/space complexity analysis of the ***SimFusion*** algorithm, are discussed below.

Two Random Walker Model and Spectral Graph Partition

Since L_{urm} can be considered as a single step transition matrix of a Markov Chain, the iterative similarity reinforcement process of Eq. (9) can be explained in a "two random walker model". Suppose two random walkers start at two data objects in the unified space and they walk from one object to another, step by step. In each step, each of them would choose the next object to set foot on according to the probability distribution of how the current data is related to other objects as defined in L_{urm}. If S^{0}_{usm} also can be rendered as an object to object relationship distribution matrix, then the reinforced similarity between the two original objects on which the two walkers started their trip, can be translated into the likelihood that the two walkers meet each other, after each of them walks *n* steps according to L_{urm}.

SimFusion also can be considered as a generic spectral graph partition process. The reasons we do not use other spectral graph partition processes such as [11] [13] are: 1) ***SimFusion*** exactly follows the Similarity Reinforcement Assumption; 2) other methods may not scale well. Detailed discussion is found in [33].

Time and Space Complexity

The space complexity of the ***SimFusion*** algorithm is $O(n^2)$ (*n* is the total number of objects in the unified space), since we only need an *n*x*n* matrix to store the ***URM*** and another *n*x*n* matrix to store the ***USM***. In each step of the reinforcement process, the similarity between two data objects *x* and *y* is updated exactly $|R(x)|+|R(y)|$ times, where $|R(x)|(|R(y)|)$ is the number of data objects to which *x(y)* relates. (Note that all 0 elements in the corresponding columns and rows in the ***URM*** and ***USM*** can be pre-excluded from the reinforcement calculation.) Suppose *d* is the average number of objects to which an object relates, then, the time complexity of the ***SimFusion*** algorithm is $O(Kn^2d)$, where *K* is the number of iterations. In the worst case, where all the data objects are fully connected (therefore *d=n*), the time complexity would increase to $O(Kn^3)$. However, in most real world scenarios, data objects are sparsely connected to each other and *d* can be considered as a constant with respect to *n*.

5.3 A Comparison with the SimRank Algorithm

Jeh and Widom proposed the ***SimRank*** algorithm [15] in 2002. In ***SimRank***, the similarity of two objects also was measured according to their contextual structure. The theoretical assumption behind the ***SimRank*** algorithm is similar to that of the ***SimFusion*** algorithm: "the similarity of two data objects can be affected by the similarities of other data objects that the two data objects are related to". The basic similarity reinforcement calculation used in ***SimRank*** is:

$$s(a,b) = \frac{C}{|R(a)||R(b)|} \sum_{i=1}^{|R(a)|} \sum_{j=1}^{|R(b)|} s(R_i(a), R_j(b)) \tag{10}$$

where *s(a,b)* is the similarity value between objects *a* and *b*; |*R(a)*| and |*R(b)*| are the total number of objects related to objects *a* and *b*, respectively. $R_i(a)$ represents the i_{th} object related to *a*. *C* is a damping factor. If we take all λs equal to 1 and average the value of relationships from one object to *1/n* (*n* is the total number of relationships from the object) in the ***URM***, then Eq. (5) can be reduced to Eq. (10) (where *C=1*). Different from the ***SimFusion*** algorithm, which uses matrices to represent object pair-wise similarities and pair-wise relationships, ***SimRank*** considered any pair of data objects as the nodes in a general directed graph, and created a directed edge from node *(a,b)* to node *(c,d)* if there are relationships from *a* to *c* and from *b* to *d*. Then, the similarity values of any nodes (pairs of data objects) are updated according to Eq. (10). The procedure discussed above is equivalent to flattening the *nxn* ***USM*** in the ***SimFusion*** algorithm into a vector of n^2 length, and then updating this vector by iteratively calculating it over a sparse n^2xn^2 matrix. ***SimRank*** also can be modeled as a modified random walker model: "Random Surfer-Pairs Model". The major differences from ***SimFusion*** are:

First, the similarity of object pair *(a,b)* is updated |*R(a)*|x|*R(b)*| times during each iteration, which is much more than the number of updates in ***SimFusion*** (|*R(a)*|+|*R(b)*|). The time complexity of ***SimRank*** is $O(Kn^2d^2)$. In the situations where data objects are heavily connected to each other (e.g., smoothing web linkage relationships [3]), the time complexity of the ***SimRank*** algorithm would grow to $O(Kn^4)$.

Second, ***SimRank*** assumes that all the relationships are binary, and Eq. (10) can be considered as taking an average of the similarity values of the object pairs that are related to object pair *(a,b)*. However, this assumption is naïve and in the real world the relationships among data objects are often unequal (e.g., a user spending more time reading web page A than web page B may indicate the user has a preference for A over B). This kind of prior knowledge can be more easily and efficiently incorporated into the ***URM*** in ***SimFusion***.

Third, the parameter *C* in ***SimRank*** is the base of the "Expected-*f* Meeting distance" function. It is difficult to understand the real world effect of *C*, and it is difficult to select *C* by intuition. However, the parameters λ_{ij} in ***SimFusion*** directly reflect the relative importance of different kinds of relationships involved in ***SimFusion*** and can be tuned by intuition. ***SimFusion*** is more flexible at combining relationships from different sources by providing a set of parameters λ_{ij}.

Table 2: A comparison of the two algorithms

Aspects	SimFusion	SimRank
Assumption	Similarity Reinforcement Assumption	A special case of the Similarity Reinforcement Assumption
Theoretical Foundation	Two random walker model	Random Surfer-Pairs Model
Time Complexity	$O(Kn^2d)$; worst case $O(Kn^3)$	$O(Kn^2d^2)$; worst case $O(Kn^4)$
Relationship Representation	Represented in values closer to the real world situations	Binary representation,
Parameter	Easy to select	Difficult to select
Real World Examples	Model most iterative/ non-iterative similarity calculating algorithms	Model a few non-iterative similarity-calculating algorithms

Fourth, ***SimFusion*** can easily be used to model most existing similarity-calculating algorithms (see Section 5.4). However, ***SimRank*** only can be used to model a few non-iterative similarity-calculating algorithms (e.g., [27]). A comparison of ***SimFusion*** and ***SimRank*** algorithms is summarized in Table 2.

5.4 Real World Examples

Simplified versions of ***SimFusion*** that only consider one or two types of data objects have been validated through prior studies. For example, consider only one data space, of journal articles, and one type of relationship, the reference relationship between journal articles. Set the initial S_{usm} to be the identity matrix. Eq. (9) reduces to co-citation [27], where the similarity of two articles is determined by the number of articles they both cite. If we only consider reverse reference relationships, Eq. (9) reduces to bibliographic coupling [17], where the similarity of two articles is determined by the number of articles that cite them both.

Let us consider the ***URM*** in Eq. (2), which represents a document space, term space, and the "containing" relationship of documents to terms. Suppose we have no prior knowledge about similarity of any data objects and set S_{usm} to be the identity matrix. Applying Eq. (9) would result in calculating the pair-wise document similarity and pair-wise term similarity according to the VSM. It remains an interesting problem, whether enriching the ***URM*** and ***USM*** with some prior knowledge (e.g., thesaurus, or document references relationships) and iteratively reinforcing the similarity, would result in better knowledge of the term similarities, document similarities, and document-term similarities?

Recently, researchers have tried to use query-web page relationships to better predict the web object similarities so as to help improve the effectiveness of web-clustering algorithms [22][29]. Their methods for calculating the similarity of web objects also can be well modeled by our ***SimFusion*** algorithm. Suppose there are two data spaces: the Web pages space and the query space. The two spaces are modeled in a ***URM*** as:

$$L_{urm} = \begin{vmatrix} \lambda_{11}L_{query} & \lambda_{12}L_{query-page} \\ \lambda_{21}L_{page-query} & \lambda_{22}L_{webpage} \end{vmatrix} \tag{11}$$

where L_{query} refers to the query content similarity relationship matrix and $L_{webpage}$ refers to the Web page content similarity relationship matrix. If $L_{query-page}$ refers to the query with its corresponding search list relationship, and L_{usm} is the identity matrix, $\lambda_{11}=\lambda_{22}=0$, and $\lambda_{12}=\lambda_{21}=1$, then applying Eq. (9) to this ***URM*** will result in Raghavan and Sever's work [22], in which they measure the similarity of queries based on corresponding result document lists. If $L_{query-page}$ refers to the web query web page click-through relationship, and S_{usm} is the identity matrix, and all the λs remain the same, applying Eq. (9) to this ***URM*** will result in Beeferman and Berger's clustering method [1], in which they measure the similarity of queries using the similarity of their clicked web pages and calculate the similarity of web pages using the similarity of the queries that lead to the selection of the web pages. If we define $\lambda_{11}>0$, $\lambda_{22}>0$, $\lambda_{12}>0$ and $\lambda_{21}>0$, applying Eq. (9) to this ***URM*** would result in Wen's work [29], where query similarity is based on both the query contents similarity and the similarity relationship of the documents that are selected by users who submitted the queries.

6. EXPERIMENTS

In this section, we show how ***SimFusion*** can be validated on a real world data set.

6.1 Design

Our experiment is designed around a real user search click-through log collected from a large scale search engine. The log contains 62.5 million query request records with the URLs of the corresponding clicked web pages from a 3 hour period in 2003. The log is formatted in such a way that each query is followed by the *URL*s of the corresponding clicked web pages and the number of clicks during a period of time, as shown below:

Query	URL	clicks	URL	clicks
Airlines	www.united.com	3452	www.aa.com	2179

We selected the top 10K popular queries in this query log and crawled all the corresponding clicked web pages; we use the top 20K popular web pages in our experiments. Then, we parsed the hyperlinks in the content of the web pages and built a hyperlink graph of the web page collection.

- The similarity of two queries can be reinforced by the similarity of the web pages they relate to.
- The similarity of two web pages can be reinforced by the similarity of the queries for which they are retrieved, as well as the similarity of other web pages to which they relate.

Thus, the web pages and the queries each form a unique data space. Queries are connected to the web pages via click-through relationships (inter-type relationship). Web pages are connected via hyperlinks (intra-type relationship) in the web page space. A ***URM*** for our data set can be built as:

$$L_{urm} = \begin{vmatrix} \alpha L_q & (1-\alpha)L_{qd} \\ (1-\alpha)L_{dq} & \alpha L_d \end{vmatrix} \qquad (12)$$

where L_q is the query inter-type relationship matrix. Since there is no intra-type relationship in the query space, L_q reduces to an identity matrix. L_d is the web pages hyperlink adjacency matrix. L_{qd} and L_{qd} are query click-through relationship matrices that connect queries with the web pages. α is a non-negative parameter that adjusts the relative importance of the click-through relationship to the hyperlink relationship during the similarity reinforcement process. Each sub-matrix in Eq. (12) is normalized to a row-stochastic matrix. We use an identity matrix as ***USM*** and we apply the ***SimFusion*** algorithm on Eq. (12) and Eq. (13) to iteratively calculate the similarities of queries and web pages. The performance of the ***SimFusion*** algorithm will be compared with a pure content similarity measurement (i.e., *tf*idf*) and the ***SimRank*** algorithm.

6.2 Evaluation Metric

We use ***Precision*** to measure the performance of the similarity calculation algorithm: Given an input object, ***Precision at N*** is defined as the number of similar data objects identified in the top ***N*** objects returned by the algorithm:

$$precision \quad at \quad N = \frac{\# \quad of \quad similiar \quad objects}{N} \qquad (14)$$

Ten human experts were hired to manually judge the similarity of objects returned by different algorithms. The final judgment of relevance was decided by majority vote.

6.3 Experimental Results

We set α=0.5 and developed the ***URM*** as in Eq. (12) and developed ***USM*** as in Eq. (13). Then, we iteratively calculated using the ***SimFusion*** algorithm until convergence was achieved (9 iterations in our experiment). Randomly selected sets of queries and URLs were used to evaluate the effectiveness of the ***SimFusion*** algorithm. The results are reported below.

6.3.1 Results on Similar Queries

Since it is difficult to evaluate the similarity of single word queries to other queries, we randomly chose 30 multi-word queries from the query log. We computed the precision at top 10 queries returned by the ***SimFusion***, ***SimRank,*** and ***tf*idf*** algorithms, for each of the 30 queries. Then, we compared the average precision at 10 for the three algorithms. The results are:

	SimFusion	***SimRank***	***tf*idf***
Average Precision at 10	0.640	0.563	0.383

The table shows that ***SimFusion*** achieves a 13.6% improvement over ***SimRank*** and a 67% improvement over the ***tf*idf*** algorithm in terms of precision at 10. A query-by-query breakdown analysis of ***SimFusion*** over ***SimRank*** is presented in Figure 4. The vertical axis indicates the percentage of improvement of ***SimFusion*** over ***SimRank***. It shows that in the 30 queries being analyzed, ***SimFusion*** outperformed ***SimRank*** in 14 queries, and was only slightly worse than ***SimRank*** in 6 queries. A t-test (2-tailed) for matched pairs showed that this improvement of ***SimFusion*** over ***SimRank*** is significant at the α=0.009 level.

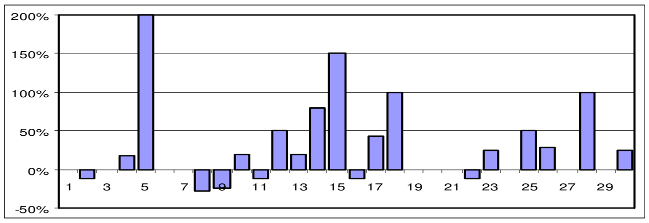

Figure 4: Query Breakdown for SimFusion vs. SimRank

To further validate our algorithm, we randomly selected another set of 20 multi-word queries and evaluated the results returned by ***SimFusion, SimRank,*** and ***tf*idf.*** The results again show that ***SimFusion*** is almost 12% better than ***SimRank*** with significance level α=0.03, and 80% better than ***tf*idf***, with significance level α=0.00002**.** A detailed report on the query sets and corresponding results in our experiments can be found in [33].

Consider a case study, analyzing the top similar queries returned for the query "pizza hut" by different algorithms (Table 3):

Bold font cells indicate similar queries. The ***tf*idf*** results are not best since they only reflect similarities in content (e.g., "pizza"). ***SimRank*** does not achieve best performance either because it returns too many semantically "marginal" relevant queries (e.g., kfc, taco bell, burger king). ***SimFusion*** acts as a combination of the other two algorithms and can achieve best performance by returning both content similar (e.g., dominos pizza) and semantically similar (e.g., papa johns, dominos) queries.

Table 3: Case study for query "pizza hut"

SimFusion	*SimRank*	*tf*idf*
pizzahut	**pizza hut**	**pizza hut**
pizza hut	**pizzahut**	**pizza**
pizza	kfc	**donatos pizza**
franchises	jack in the box	**dominos pizza**
franchise	dairy queen	N/A
kfc	kentucky fried chicken	N/A
papa johns	taco bell	N/A
dominos pizza	red lobster	N/A
dominos	burger king	N/A

We also analyze how the number of iterations can affect the performance of the ***SimFusion*** algorithm. We evaluate the precision of ***SimFusion*** at each iteration (1 to 9), and draw the precision vs. iteration curve in Figure 5. We see that ***SimFusion*** improves the precision faster at the initial iterations than at the latter iterations. Similar findings have been reported in [15].

6.3.2 Results on Similar Web pages

We use the metric of Section 6.2 to evaluate the performance of ***SimFusion*** on web pages. We randomly chose 10 web pages from the log. For each of them we compare the similarity of the top 10 web pages returned by ***SimFusion***, ***SimRank,*** and ***tf*idf***.

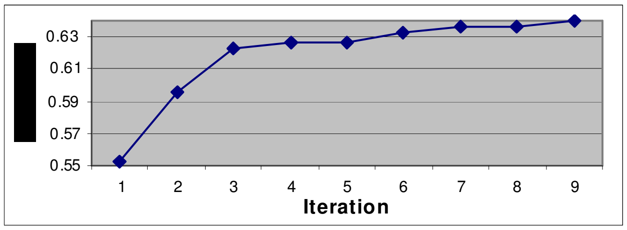

Figure 5: Precision vs. Iteration curve for SimFusion

We found that the average precision for ***SimFusion*** (0.67) is 71% better than ***SimRank*** (0.39) and 29% better than ***tf*idf*** (0.52). A 2-tailed t-test shows that improvement is significant at α=0.005 versus both ***SimRank*** and ***tf*idf***. A detailed page-by-page comparison of the three algorithms is shown in Figure 6. Further information can be found in **Error! Reference source not found.**.

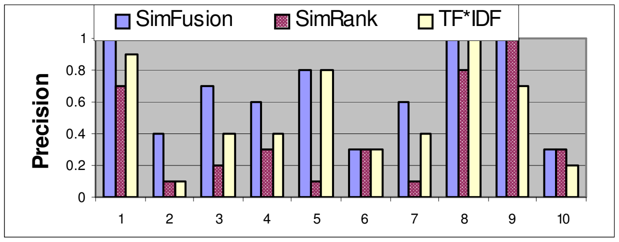

Figure 6: SimFusion vs. SimRank on web page similarities

To further validate our algorithm, we randomly selected another set of 10 web pages and evaluated the similar web pages returned by ***SimFusion***, ***SimRank,*** and ***tf*idf.*** The results again show that the performance of ***SimFusion*** (0.64) is 110% better than ***SimRank*** (0.3), with significance level α=0.003, and is 64% better than ***tf*idf*** (0.39), with significance level α=0.006 (2 tailed t-tests).

We also conducted a case study (see Table 4) by analyzing the top similar web pages for "http://www.tvguide.com", as returned by two algorithms. Bold font cells indicate similar web pages. We see that ***SimFusion*** has returned one more semantically similar web page (e.g., http://tvlistings2.zap2it.com) than ***tf*idf*** while keeping other content similar web pages. On the other hand ***tf*idf*** returns some non-similar web pages on top (e.g., www.nbc.com), since that sites' topmost dynamic page just happened to advertise the same TV shows as the target page, at the time we crawled the web pages.

Table 4: Case study for web page "www.tvguide.com"

SimFusion	*tf*idf*
http://tv.msn.com/default.aspx	**http://www.tvguide.com**
http://www.tvguide.com	**http://tv.msn.com/default.aspx**
http://tvlistings2.zap2it.com	http://nbc.com/nbc/header/local_stations
http://www.worldnetdaily.com	http://www.nbc.com
http://www.fcc.gov/dtv	http://www.nbc.com/nbc/last_comic_standing:_the_search_for_the_funniest_person_in_america/contestants/dat_phan.shtml
http://groups.msn.com/browse?catid=65	http://www.sirlinksalot.net/paradisehotel.html

6.3.3 Parameter selection

We investigated how different values of α defined in Eq. (12) can effect the performance of the ***SimFusion*** algorithm. We chose 10 α values from 0 to 1, at intervals of 0.1, and evaluated the performance of ***SimFusion*** on finding similar queries at each α value. Figure 7 illustrates the resulting performance curve.

We see that the ***SimFusion*** algorithm achieves best performance when α=0.3, which indicates that the query web page click-through relationship is more important than the content similarity of queries, when used to calculate the similarity of different queries. When analyzing the data, we also found that different queries find their most similar queries at different α values. Might it be possible to automatically determine a set of parameters for each data object being considered in the ***URM****?*

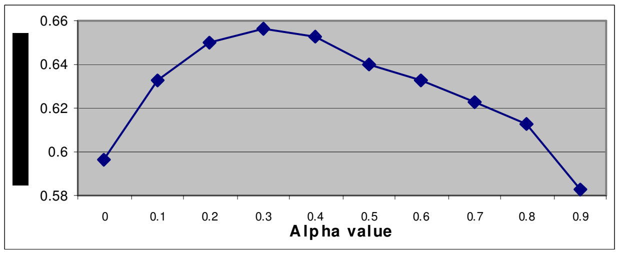

Figure 7: Performance curve for different α values

7. CONCLUSION AND FUTURE WORK

In this paper, we introduce the Unified Relationship Matrix (***URM***) to represent heterogeneous data objects and their relationships in a unified manner. We addressed the information integration problem by showing how different relationships can be used to improve the similarity measurement of data objects. Next, we introduced the ***SimFusion*** algorithm. By iteratively computing over the ***URM***, the ***SimFusion*** algorithm can effectively integrate relationships from multiple sources to measure the similarity of data objects. Experiments based on real world data demonstrate that the ***SimFusion*** algorithm can significantly improve the similarity measurement of data objects over both the traditional content-based algorithms and the cutting edge ***SimRank*** algorithm.

In the future, we plan to use machine learning technologies to automatically optimize the parameters used in ***SimFusion***. We also will try to improve the efficiency of ***SimFusion*** so that it can be easily applied to large scale mining applications (e.g., MapReduce [9]).

8. REFERENCES

[1] D. Beeferman and A. Berger. "Agglomerative clustering of a search engine query log", *in Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, pp. 407-415, Aug. 2000.

[2] T. L. Brauen, "Document Vector Modification", in *The Smart Retrieval System-Experiments in Automatic Document Processing*, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, Chapter 24, 1971.

[3] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30, pp. 107-117, 1998.

[4] V. Bush, "As We May Think", *The Atlantic Monthly*, vol. 176, pp.101-108, July 1945.

[5] P. Calado and B. Ribeiro-Neto, "An Information Retrieval Approach for Approximate Queries," *IEEE Transactions on Knowledge and Data Engineering*, 15: 236-239, 2003.

[6] S. Chakrabarti, B.E. Dom, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. M. Kleinberg, “Mining the Web's Link Structure”. *IEEE Computer*, 32 (8)., pp. 60-67, 1999.

[7] G. Das, H. Mannila, P. Ronkainen, “Similarity of attributes by external probes”, *in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 23–29, 1998.

[8] B. D. Davison, “Toward a unification of text and link analysis.” *in Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval,* Toronto, Canada, pp. 367-368. 2003.

[9] J. Dean and S. Ghemawat, “ MapReduce: Simplified Data Processing on Large Clusters”, *in Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI'04),* San Francisco, CA, pp. 137-150, Dec. 2004.

[10] J. Dean and M.R. Henzinger. “Finding Related Pages in the World Wide Web”, *in Proceedings of the 8th international conference on World Wide Web*, 1999.

[11] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, R. Harshman "Using latent semantic analysis to improve information retrieval." In *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 281-285, Washington D.C., May, 1988.

[12] N. Fuhr and T. Rolleke, “A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems,” *ACM Transactions on Information Systems*, vol. 15, pp. 32-66, 1997.

[13] D. Gibson, J. Kleinberg, and P. Ragavan, “Clustering Categorical Data: An Approach Based on Dynamic Systems”, in *Proceedings of the 24th International Conference on Very Large Databases*, 1998.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering”, *in Proc. 22nd annual international ACM SIGIR conference on research and development in information retrieval*, pp. 230-237, Berkeley, California, July, 1999.

[15] J. Jeh and J. Widom. “SimRank: a measure of structural-context similarity”. *In Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining,* pp. 538-543. Edmonton, Alberta, Canada, July 23-26, 2002.

[16] O. Kallenberg, *Foundations of Modern Probability.* New York: Springer-Verlag, 1997.

[17] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10-25, 1963.

[18] J.M. Kleinberg, “Authoritative Sources in a Hyperlinked Environment”. *JACM*, 46 (5), pp. 604-632, 1999.

[19] R.R. Larson. “Bibliometrics of the World-Wide Web: An exploratory analysis of the intellectual structure of cyberspace”, *in Proceedings of the Annual Meeting of the American Society for Information Science*. Baltimore, Maryland, October, 1996.

[20] N. Liu et al. “A similarity reinforcement algorithm for heterogeneous Web Pages”, *in Proceedings of the seventh Asia Pacific Web Conference*, Shanghai, March, 2005.

[21] A. Popescul, G. Flake, S. Lawrence, L.H. Ungar, and C.L. Giles. “Clustering and identifying temporal trends in document database”, *in Proceedings of the IEEE Advances in Digital Libraries*, Washington, D.C., May, 2000.

[22] V.V. Raghavan and H. Sever. “On the reuse of past optimal queries”, *in Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, WA. pp. 344-350, July, 1995.

[23] P. Resnick, and H. R. Varian, “Recommender Systems” (*introduction to special section*). *Communications of the ACM*, 40(3):56-58, March, 1997.

[24] B. Ribeiro-Neto and R. Muntz, “A Belief Network Model for IR”, in *Proceedings of the 19th ACM-SIGIR conference on research and development in information retrieval*, pp. 253-260, Zurich, Switzerland, 1996.

[25] J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System - Experiments in Automatic Document Processing.* Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[26] G. Salton, *Automatic Information Organization and Retrieval*, McGraw-Hill, 1968.

[27] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents, *J. of the American Society of Information Science* 24:265-269, 1973.

[28] H. Turtle and W. B. Croft, “Inference networks for document retrieval”, *in Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Brussels, Belgium, 1990.

[29] J.-R.Wen, J.-Y. Nie, and H.-J. Zhang, “Query Clustering Using User Logs”, *ACM Transactions on Information Systems (TOIS)*, 20 (1): 59-81.

[30] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong, “On Modeling of Information Retrieval Concepts in Vector Space”, *ACM TODS*, 12: 299-321, 1987.

[31] W. Xi, et al.,“Link Fusion: A Unified Link Analysis Framework for Multi-type Inter-related Data Objects”, in *Proceedings of the 13th International World Wide Web Conference,* pp.319-327*,* New York, May 2004.

[32] G. Xue, et al., “MRSSA: An Iterative Algorithm for Similarity Spreading over Interrelated Objects”, in *Proceedings of the 13th Conference on Information and Knowledge Management (CIKM2004)*, pp. 240-241, Washington, D.C., Nov, 2004.

[33] W. Xi, B. Zhang and E. A. Fox “SimFusion, A Unified Similarity Measurement Algorithm for Multi-type Interrelated Web Objects”, *Technical Report, TR-04-19, Computer Science Department, Virginia Tech,* Dec. 2004.