SimFusion: A Unified Similarity Measurement Algorithm for Multi-Type Interrelated Web Objects

Wensi Xi
Virginia Tech
Blacksburg, VA, 24061
xwensi@vt.edu

Benyu Zhang
Microsoft Research Asia
Beijing, China, 100080
byzhang@microsoft.com

Edward A. Fox
Virginia Tech
Blacksburg, VA, 24061
fox@vt.edu

ABSTRACT
In this paper, we use a Unified Relationship Matrix (***URM***) to represent a set of heterogeneous web objects (e.g., web pages, queries) and their interrelationships (e.g., hyperlink, user click-through relationships). We claim that iterative computations over the ***URM*** can help overcome the data sparseness problem (a common situation in the Web) and detect latent relationships among heterogeneous web objects, thus, can improve the quality of various information applications that require the combination of information from heterogeneous sources. To support our claim, we further propose a unified similarity-calculating algorithm, the ***SimFusion*** algorithm. By iteratively computing over the ***URM***, the ***SimFusion*** algorithm can effectively integrate relationships from heterogeneous sources when measuring the similarity of two web objects. Experiments based on a real search engine query log and a large real web page collection demonstrate that the ***SimFusion*** algorithm can significantly improve similarity measurement of web objects over both traditional content based similarity-calculating 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, Similarity-calculating algorithms, Information retrieval, Data fusion, Data mining.

1. PROBLEM CONTEXT
Web pages, *users*, *queries* and other entities in the web domain can be considered as objects containing information. The information may include the content of objects as well as relationships, between objects of the same or different types. For example, we know that *users* browse *web pages* and issue *queries*. *Queries* lead to the click-through of *web pages*. These three operations (browsing, issuing, and clicking-through) 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 latter connections are all examples of relationships within the same type of objects.

Modern information applications such as searching, document clustering, and collaborative filtering, use three traditional approaches to represent information derived from data objects and the interrelationships involved:

1. Spaces: Vector and probability spaces implicitly use a pre-defined set of features from objects, e.g., to locate them in an n-dimensional space [28][34].
2. Databases: Relational databases operate on dynamically specified relationships between objects (each represented using a pre-defined set of attributes) [5][12].
3. Networks: Belief, inference, and spreading activation networks (e.g., neural net) use nodes and arcs to connect objects and their attribute values, and to represent pre-defined sets of relationships [1][25][31].

However, most information applications simply take one of these three approaches to analyze only one kind of relationship within the same type of objects (e.g., document clustering) or between two types of objects (e.g., searching, collaborative filtering). These applications, as how implemented, will run into problems when users require more accurate models of reality, wherein the number of types of objects that must be handled in an integrated manner expand rapidly (e.g., considering both *queries* and *users* when clustering *web pages*), and the relationships between different types of objects (e.g., the reference and browsing relationships among *web pages* and between *users* and *web pages*) grow tremendously. More specifically, the problem we are facing can be described as:

"How can the broad variety of heterogeneous data and relationships in the Web be effectively integrated to improve the performance of diverse information applications?"

Our hypothesis is that the quality and utility (e.g., the similarity value) of information contained in web objects can be improved by integrating interrelationships of related objects from different types of sources. Since relationships among data objects often can be represented as matrices with either binary or real-valued weights (e.g., a web hyperlink adjacency matrix), such an integration process can be modeled as calculations over relationship matrices. Figure 1 illustrates a few examples of such calculations. Note that as a result of integration calculations, the matrices in the bottom row, which can be used to explain some possible information applications, are presumed to be of higher quality (e.g., with less empty spaces, due to the addition of new relationships obtained from other matrices). A way to think of why different matrices of relationships can be used to improve the quality of information is that a single matrix representation of relationships is often very sparse, but when reinforced by other types of relationship matrices, the information it contains may be more dense and helpful. The underlying assumption is that the relationship of two data objects can be affected by similar

WWW 2005, May 10--14, 2005, Chiba, Japan.

relationships of data objects they are related to from among multiple data spaces, through varied interrelationships.

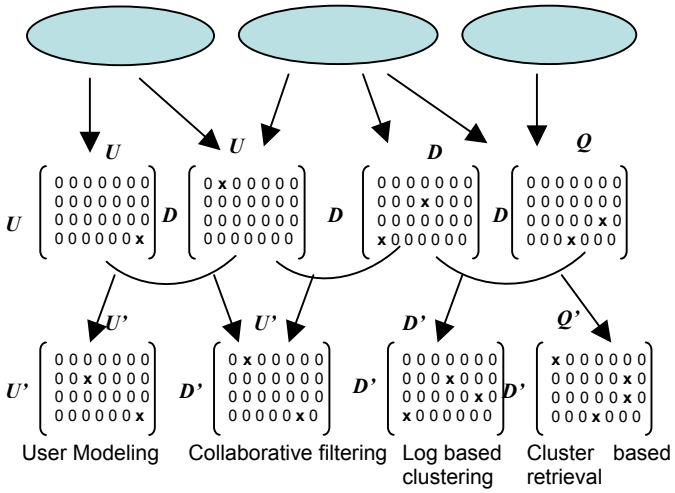

Figure 1: Matrix representations of relationship integration

In this paper, we tackle the bigger problem of integrating multiple relationships to improve information applications by analyzing a specific question: "How can different relationships among data objects be used to improve the measurement of a specific relationship, namely the *similarity relationship* of data objects?" In this work, we solve this problem by introducing a novel algorithm, ***SimFusion***, which iteratively integrates relationships from multiples sources to improve the quality of similarity calculation over web objects. The effectiveness of the ***SimFusion*** algorithm will be tested on a real web data set.

The rest of this paper is organized as follows. In Section 2 we explain the background of our research by reviewing some of the large number of previous works that relate to information integration. In Section 3 we give the formal definition of the Unified Relationship Matrix (***URM***) and explain other terminology. The underlying assumption and formal description of the ***SimFusion*** algorithm is presented in Section 4. Then, we give experimental results in Section 5 and conclude in Section 6.

2. LITERATURE REVIEW

In this work, we show that different kinds of relationships can be integrated to improve the similarity measurement of web objects. Thus it is helpful to trace the evolution of how relationships are used to measure the similarity of data objects in various information applications such as searching and clustering.

Most early research studies considered a single relationship to measure the similarity of data objects. In the vector space model [27], terms were used to characterize queries and documents, creating a document-term relationship matrix where it is straightforward to compute the similarities between terms and documents by taking the inner product of the two corresponding row or column vectors. Dice, Jaccard, and Cosine [23] are classical methods that use the document-term relationship to measure the similarity of documents for retrieval and clustering purposes. Deerwester and Dumais [9][10] demonstrated that a same concept might be presented by different sets of keywords in different documents. In their Latent Semantic Indexing work, instead of directly using the document-term matrix to compute the similarity of text objects, they first use the Singular Vector Decomposition to map the document-term matrix into a lower dimension matrix where each dimension is associated with a set of keywords and with a "hidden" concept. Then the similarity of text objects (documents or queries) is measured by their relationships to these "concepts" rather than the keywords they contain.

Other single type relationships such as reference relationships among scientific articles also are used to measure the similarity of data objects. Small [29] measured the similarity of two journals by counting the number of papers they both cite; this is called co-citation. Kessler [15] measured the similarity of two papers by counting the number of papers that cite them both; this is called bibliographic coupling. Co-citation and bibliographic coupling have been successfully used to cluster scientific journals [21]. With the advent of the World Wide Web, relationships within web objects (e.g., hyperlinks) also were used to calculate the similarity of web objects. Dean [8] and Kleinberg [16] used hyperlinks in the Web to discover similar web pages. Larson [17] and Pitkow [20] applied co-citation on the hyperlink structure of the web to measure the similarity of web pages. In the Collaborative Filtering [18] and Recommender Systems [24] area, researchers tried to analyze the similarity of people by examining the people-document and people-artifact relationships, respectively. A few examples of applications that make use of relationships to calculate the similarity of data objects are shown in Table 1.

Table 1. Some relationships in information applications

Information Application	Modeling Relationship
Information retrieval	Term-document relationship
Document clustering	Term-document relationship or Document-document relationship
Collaborative filtering	People-document relationship
Recommending	People-artifact relationship

The works introduced above only used a single type relationship to measure the similarity of data objects. However, these approaches run into serious problems when various information applications require a more real and accurate similarity measuring method where multiple types of data objects and their relationships must be handled in an integrated manner. Thus in the extended VSM [11], feature vectors of data objects were lengthened by adding attributes from objects of other related types via inter-type relationships. By doing so, information from different sources are directly mapped into an enhanced Vector Space and similarity computations were obtained by calculating these enhanced feature vectors. The extended feature vectors were used for document search and clustering purposes [6]. Following the same idea, Rocchio [26] and Ide [14] expanded query vectors using the terms appearing in the top documents retrieved by the query and improved the search effectiveness. The idea of using terms found in related documents to extend the query vector is also referred to as "Query Expansion". Similarly, Brauen [3] modified a document vector by adding or deleting the terms in the queries that relate to it. Changing document vectors by related query terms is also referred to as the "Dynamic Document Space" technique [28].

Recently, researchers have tried to calculate the similarity of same type data objects by measuring their relationships across different types of data objects. For example, Raghavan and Sever [22] tried to measure the similarity of two queries by calculating the similarity relationship of their corresponding search lists.

Beeferman and Berger [2] clustered queries using the similarity of their clicked web pages and cluster web pages using the similarity of the queries that lead to them. Wen [32] and Su [30] calculated the query similarity based on both the query contents similarity and the similarity of documents retrieved by the queries; they calculated the similarity of documents in a similar way.

Although research works introduced above used inter-type relationships to help improve the similarity measurement of data objects, they did not consider the mutual reinforcement effect on the relationships across multi-type data objects. Wang et al. [33] proposed an iterative reinforcement clustering algorithm for multi-type data objects named ***ReCom***. In this algorithm, the clustering results from one type of data object are used to reinforce the clustering process of another data type. Their method was shown to be effective and can be considered as a variation of the ***SimFusion*** algorithm that we introduce later in this work.

Davidson [7] proposed another related idea. He analyzed multiple term document relationships by expanding the traditional document-term matrix into a matrix with term-term, doc-doc, term-doc, and doc-term sub-matrices in a way that is similar to the unified relationship matrix (***URM***) that we define later. He proposed that the similarities of the search objects (web pages or terms) in the expanded matrix could be emphasized. With enough emphasis, the principal eigenvector of the extended matrix will have the search object on top with the remaining objects ordered according to their relevance to the search object. Although his idea is sound, the iterative calculation over the ***URM*** for related web pages would make it prohibitive as an online algorithm.

In this work, we introduce the ***SimFusion*** algorithm, which iteratively updates the similarity of data objects via multiple interrelationships from heterogeneous data spaces. The most similar work in literature so far is the ***SimRank*** algorithm proposed by Jeh and Widom [13]. In ***SimRank***, the similarity of data objects was measured according to their structural context. The basic rationale of ***SimRank*** was that the similarity of two data objects could be affected by the similarities of other data objects that the two data objects related to. Jeh and Widom considered all the pair-wise similarities of data objects as nodes in a general directed graph, and mapped all the contextual relationship into directed edges in the graph. They then iteratively updated the similarity of data object pairs in a manner similar to the PageRank [4] algorithm. Xue et al. applied ***SimRank*** on the web domain to improve the similarity calculation of web pages and queries [36]. They also tried to calculate the similarities of queries and web pages and append query contents to most similar web pages so as to effectively improve search performance [37].

The ***SimFusion*** algorithm we propose in this paper arises from similar rationale as the ***SimRank*** algorithm. However, we claim that the basic assumption of the ***SimRank*** algorithm can be regarded as a special case of the assumption in the ***SimFusion*** algorithm. We further argue that the ***SimFusion*** algorithm has a more solid theoretical foundation, lower time complexity, and is a more flexible to be adapted into real world scenarios than ***SimRank***. A detailed comparison of the two algorithms can be found in Section 4.

3. REPRESENTATION

3.1 Terminology

It is important to first give the formal definitions of some key terms that will be used extensively in the rest of this paper.

Data Type: A **data type** is defined as a set of characteristic features (e.g., user is a set of features including name, gender, age, education, etc.). A **data object** is an instance of a data type.

Data Space: A **data space** is a set of **data objects** of the same data type (e.g., the *web pages* in the internet.).

Homogeneous/Heterogeneous: In this work we assume that each data space is **homogeneous** within itself, but **heterogeneous** with respect to other data spaces of different data types.

According to their types, relationships among data objects can be classified into two types: **intra-type relationship** and **inter-type relationship**.

Intra-type relationship: this kind of relationship connects data objects within a homogeneous data space (e.g., the hyperlinks in web pages space can be considered as intra-type relationships).

Inter-type relationship: this kind of relationship connects data 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, or the user and query data spaces, respectively).

In order to illustrate all the terminologies introduced above, we give an example with two data types: people and documents. Authors are data objects in the "people" space. Co-authorship is an intra-type relationship connecting two authors. Authorship of a document is an inter-type relationship between an "author" object and a "document" object. Readers are also objects in the "people" space, and readers reading papers can be regarded as another inter-type relationship that connects the "people" space and the "document" space. Papers in the document space may also relate to each other via some intra-type relationship (such as reference).

3.2 Unified Relationship Matrix and Examples

Below is the formal definition of the Unified Relationship Matrix that represents both inter- and intra-type relationships among data objects from heterogeneous sources in a unified manner.

Suppose we have t different data spaces $S_1, S_2, \ldots S_t$. 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), where l_{xy} represents the inter-type relationship from the xth object to the yth 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 l_{xy} represents the inter-type relationship from the xth object in S_i to the yth object in S_j.

To simplify the problem, let's first consider two data spaces $X = \{x_1, x_2, \cdots x_m\}$, and $Y = \{y_1, y_2, \cdots y_n\}$ and their relationships: R_x, R_y, R_{xy}, and R_{yx}. The adjacency matrices L_x and L_y stand for the intra-type relationship within the data spaces X and Y, respectively. L_{xy} and L_{yx} are the inter-type relationships from objects in X to objects in Y and inter-type relationships from

objects in Y to objects in X respectively. If data spaces X and Y are merged into a unified data space U, then, previous inter- and intra-type relationships are now all part of intra-type relationships R_u in U. Suppose L_u is the adjacency matrix of R_u, then L_u is a $(m+n)\times(m+n)$ matrix, with l_{ij} representing the relationship from the *ith* object from X (if $i \le m$), or the *(i-m)th* object from Y (if $i > m$), to the *jth* object from X (if $i \le m$), or the *(j-m)th* object from Y (if $i > m$). The Unified Relationship Matrix L_u is actually a matrix that combines L_x, L_y, L_{xy} and L_{yx} as shown below:

$$L_u = \begin{vmatrix} L_x & L_{xy} \\ L_{yx} & L_y \end{vmatrix} \quad (1)$$

Eq. (1) can easily be extended to the definition of the Unified Relationship Matrix L_{urm} for N data spaces, as shown in Eq. (2).

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

For the rest of this proposal I will use ***URM*** to denote the Unified Relationship Matrix.

The ***URM*** can be used to explain many real world information applications. For example, if we only consider one data space: the web pages, and one type of intra-type relationship: the hyperlinks, the ***URM*** is reduced to the link adjacency matrix of the web graph. If we analyze how user-browsing behaviors can affect the "popularity" of a web page as in [4], we would be actually analyzing two data spaces: user and web page, as well as one inter- (browsing) and two intra- (hyperlink, user endorsement) type relationships.

If we consider two data spaces: documents and terms, the 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 below.

$$L_{urm} = \begin{vmatrix} 0 & L_{dt} \\ L^T{}_{dt} & 0 \end{vmatrix} \quad (3)$$

L_{dt} is the traditional document-term matrix that represents the Vector Space Model [27]. The 0 sub-matrices in the diagonal direction indicate that we have no prior knowledge of intra-type relationships within documents and 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 space 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 Vector Space 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 first suggested by Davidson [7] as the "generic augmented matrix".

The ***URM*** introduced in this section has provided a more generalized way of viewing heterogeneous data objects and their relationships. In the ***URM***, different types of data objects are treated as elements of a "unified" data space. Previous inter- and intra-type relationships are now considered as a generic intra-type relationship that connects data objects in the "unified" space. Current information applications that measure data object information/relationships using a single type relationship matrix (e.g., document similarity calculation) can be extended and applied on the ***URM***. If designed properly, these extended applications would out-perform in effectiveness their traditional counterparts, because they can use multiple types of relationships to improve the measure of the target information/relationship.

4. ALGORITHM

4.1 Assumption

In this section, we analyze how different kinds of relationships among heterogeneous data objects can be used to reinforce a specific relationship, the similarity relationship of data objects. The underlying 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". It is named "similarity reinforcement assumption" and is illustrated below:

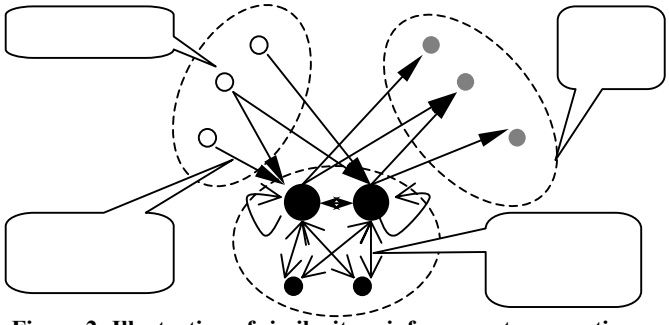

Figure 2: Illustration of similarity reinforcement assumption

In Figure 2, the similarity between two data objects (big black nodes) was 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 and $S_{ij}(x,y)$ is the similarity value 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:

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

where, α and β are positive parameters used to adjust 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 during the reinforcement process. 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:

$$S_{ij}^{new}(x,y) = \lambda_{ii} R_{ii}(x,x) \lambda_{jj} R_{jj}(y,y) S_{ij}^{original}(x,y) + \sum_{\forall a \in \forall k, \forall b \in \forall l} \lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}^{original}(a,b) \quad (5)$$

Please note that in Eq.(5), a can not be equal to x and b can not equal to y at the same time in $\lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}(a,b)$. Further, $\lambda_{ii} R_{ii}(x,x) \lambda_{jj} R_{jj}(y,y) S_{ij}^{original}(x,y)$ can be considered as a special case of $\lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}(a,b)$, where $a=x$ and $b=y$. Thus, Eq. (5) can be further reduced to Eq. (6):

$$S_{ij}^{new}(x,y) = \sum_{\forall a, \forall b} \lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}^{original}(a,b) \quad (6)$$

Considering one data object's related objects in other data spaces as its mappings in those data spaces, the reason that similarity reinforcement process can better predict the similarity of two data objects is that the similarity of two data objects is measured in multiple perspectives (data spaces) instead of single perspective. Explained in a more easily understood fashion: two men are more likely to be "good friends" if their wives are good friends too and their children go to the same school. It should be noted that the underlying assumption is that relationships are accurate and additive. Thus, care should be taken to avoid situations where there are contradictory or ambiguous types of evidence.

4.2 The SimFusion Algorithm

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

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. (7):

$$L_{urm} = \begin{vmatrix} \lambda_{11}L_{11} & \lambda_{12}L_{12} & \cdots & \lambda_{1N}L_{1N} \\ \lambda_{21}L_{21} & \lambda_{22}L_{22} & \cdots & \lambda_{2N}L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N1}L_{N1} & \lambda_{N2}L_{N2} & \cdots & \lambda_{NN}L_{NN} \end{vmatrix} \quad (7)$$

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 ith row of the matrix L_{ij} are zero), then each element in the ith row of relationship matrix L_{ij} is set to $1/n$, where n is the 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, $\sum_{\forall j} \lambda_{ij} = 1 \quad and \quad \forall i,j \quad \lambda_{ij} > 0$. Thus, Eq. (7) is a row-stochastic matrix and can be rendered as a single step probability transformation matrix in a Markov Chain [19].

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, as shown in Eq. (8):

$$S_{usm} = \begin{vmatrix} 1 & s_{12} & \cdots & s_{1T} \\ s_{21} & 1 & \cdots & s_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1T} & s_{2T} & \cdots & 1 \end{vmatrix} \quad (8)$$

Each element $s_{(a,b)}$ in S_{usm} represent the similarity value between data object 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 $0 \le s_{ab} \le 1$, if $a \ne b$. S_{usm} is a symmetric matrix since $s_{ab} = s_{ba}$. We also define that the orders 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 object a and b. Having ***URM*** and ***USM*** defined, the similarity reinforcement assumption can be represented as:

$$S_{usm}^{new} = L_{urm} S_{usm}^{original} L_{urm}^{T} \quad (9)$$

Eq. (9) is the basic similarity reinforcement calculation in the ***SimFusion*** algorithm. Eq. (9) be continued in an iterative manner until the calculation converges or a satisfaction result is obtained, as shown in Eq. (10).

$$S_{usm}^{n} = L_{urm} S_{usm}^{n-1} L_{urm}^{T} = L_{urm}^{n} S_{usm}^{0} (L_{urm}^{n})^{T} \quad (10)$$

The proof of convergence for Eq. (10) can be found in the Appendix. It 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. (10) may not be equal to 1 and may even be smaller than the similarity between two data objects (i.e., values in non-diagonal positions in S^{n}_{usm}). However, we argue that the S_{usm} derived during the iterative calculation can be rendered more precisely as the confidence of the similarity of individual or pairs of data objects rather than as their 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 (confident) 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 below:

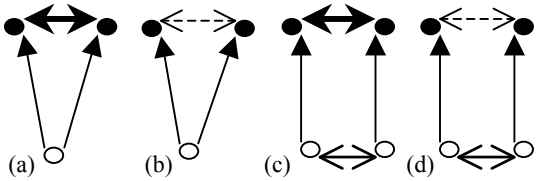

Figure 3: Illustration of similarity confidence for single object and object pairs

In Figure 3, the white nodes at the bottom represent the object 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. According to our discussion, (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 object pair.

The theoretical foundation, time/space complexity, and extensions of the ***SimFusion*** algorithm will be discussed in the following.

Two Random Walker Model

Since L_{urm} can be considered as a single step transition matrix of a Markov Chain, the iterative similarity reinforcement process of Eq. (10) 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 both of them walk n steps according to L_{urm}, and then, either one of them takes a final step according to S_{usm}.

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), because we only need an nxn matrix to store the ***URM*** and another nxn 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 that $x(y)$ relates to (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 that an object relates to, then, the time complexity of the ***SimFusion*** algorithm is $O(Kn^2d)$, where K is the number of iterations. In the worst case that 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.

Extensions of SimFusion Algorithm

The similarity of data objects can not only be reinforced by the relationships that they lead to, but also can be reinforced by the relationships that lead to them (e.g., [15]). If we create a ***URM*** to represent all the inbound relationships in a similar way to the ***URM*** for outbound relationships, then Eq. (9) can be expanded to incorporate the similarity reinforcement from both inbound relationships and outbound relationships as shown in Eq. (11).

$$S^n_{usm} = \alpha L^{out}_{urm} S^{n-1}_{usm} (L^{out}_{urm})^T + (1-\alpha) L^{in}_{urm} S^{n-1}_{usm} (L^{in}_{urm})^T \qquad (11)$$

In Eq. (11), where L^{out}_{urm} is the ***URM*** for all the outbound relationships, then $\alpha L^{out}_{urm} S^{n-1}_{usm} (L^{out}_{urm})^T$ can be considered as the ***forward similarity reinforcement component***, where the similarity between data objects is reinforced by outbound relationships. L^{in}_{urm} is the ***URM*** for all the inbound relationships and $(1-\alpha) L^{in}_{urm} S^{n-1}_{usm} (L^{in}_{urm})^T$ can be considered as the ***reverse similarity reinforcement component***, where the similarity between data objects is reinforced by the inbound relationships. α is a nonnegative parameter used to adjust the relative importance between the forward and the reverse similarity reinforcement components. The proof of convergence for Eq. (11) also can be found in the Appendix.

The ***SimRank*** algorithm can be modified slightly so that after each reinforcement iteration, data objects in some spaces are grouped into clusters according to some clustering algorithm, and the corresponding relationship matrices in the ***URM*** and similarity matrices in ***USM*** also are reduced to cluster-cluster and cluster-object relationship/similarity matrices. The modified ***URM*** and ***USM*** then can be used in the next iteration of the similarity reinforcement calculation. The similarity values calculated are used again to cluster data objects in different data spaces. This modified iterative reinforcement process can be considered as an extension of the ***ReCom*** algorithm [33] that iteratively clusters data objects in different data spaces using interrelationships.

4.3 A Comparison with the SimRank Algorithm

Jeh and Widom proposed the ***SimRank*** algorithm [13] in 2002. In ***SimRank***, the similarity of two objects also was measured according to their contextual structure (relationships to other objects). 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 the ***SimRank*** is:

$$s(a,b) = \frac{C}{|R(a)||R(b)|} \sum_{i=1}^{|R(a)|} \sum_{j=1}^{|R(b)|} s(R_i(a), R_j(b)) \qquad (12)$$

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 ith object related to a. C is a dampening factor. If we take $C=1$ 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. (6) can be considered as a special case of Eq. (12). Different from the ***SimFusion*** algorithm, which uses matrices to represent object pair-wise similarities and pair-wise relationships, the ***SimRank*** algorithm 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. (12) in a similar fashion as in the PageRank algorithm. 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 n^2 length vector by iteratively calculating it over a sparse n^2xn^2 matrix. Jeh and Widom interpret their algorithm as a modified random walker model: "Random Surfer-Pairs Model".

There are several major differences between the ***SimRank*** and ***SimFusion*** algorithms:

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 the ***SimFusion*** algorithm ($|R(a)|+|R(b)|$). The time complexity of the ***SimRank*** algorithm is $O(Kn^2d^2)$ where K is the number of iterations and d is the average number of relationships a data object has in the unified data space. In the situations where data objects are heavily connected to each other (e.g., smooth web linkage relationships [4]), the time complexity of the ***SimRank*** algorithm would grow to $O(Kn^4)$.

Second, the ***SimRank*** algorithm assumes that all the relationships are binary, and Eq. (12) 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 too naïve, since 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 rather than for B). This kind of prior knowledge can be more easily and efficiently incorporated into the ***URM*** in the ***SimFusion*** algorithm.

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

Fourth, the ***SimFusion*** algorithm can easily be used to model most existing similarity-calculating algorithms as described in Section 4.4. However the ***SimRank*** algorithm only can be used to model a few non-iterative similarity-calculating algorithms (e.g., [29]). A comparison of the ***SimFusion*** and ***SimRank*** algorithm is summarized in the table below:

Table 2. A comparison of the two algorithms

Aspects	SimFusion Algorithm	SimRank Algorithm
Assumption	Similarity Reinforcement Assumption	A special case of 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, naïvely takes the average of the relationships
Parameter Selection	Easy to comprehend and select by intuition	Difficult to select by intuition
Real World Examples	Model most existing iterative /non-iterative similarity calculating algorithms	Model only a few non-iterative similarity-calculating algorithms

4.4 Real World Examples

Simplified versions of ***SimFusion*** that only consider one or two types of data objects have been validated through varied experiments. For example, we considered only one data space, the space of journal articles, and one type of relationship: the reference relationship between journal articles, and set the initial S_{usm} as the identity matrix. Eq. (11), if we set $\alpha=1$, actually reduces to the co-citation [29] situation, where the similarity of two articles is determined by the number of articles they both cite. If we set α=0, the reduction is to the bibliographic coupling [15] situation, where the similarity of two articles is determined by the number of articles that cite them both.

Let us consider the ***URM*** in Eq. (3), 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 traditional Vector Space method. 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 such as [22][32]. 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 shown in Eq. (7).

$$L_{urm} = \begin{vmatrix} \lambda_{11}L_{query} & \lambda_{12}L_{query-page} \\ \lambda_{21}L_{page-query} & \lambda_{22}L_{webpage} \end{vmatrix} \quad (13)$$

where L_{query} refers to the query content similarity relationship matrix and $L_{webpage}$ refers to the Web page content similarity relationship matrix. If $L_{query\text{-}page}$ refers to the query with its corresponding search list relationship, and L_{usm} is the identity matrix, and $\lambda_{11}=\lambda_{22}=0$, $\lambda_{12}=\lambda_{21}=1$, then applying Eq. (16) on this ***URM*** will result in Raghavan and Sever's [22] work, in which they measure the similarity of queries based on corresponding result document lists. If $L_{query\text{-}page}$ 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. (16) on this ***URM*** will result in Beeferman and Berger's[2] clustering method, 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 this ***URM*** in Eq. (16), would result in Wen's [32] work, 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.

5. EXPERIMENTS

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

5.1 Design

Our experiment is designed around a real user search click-through log collected from the MSN search engine. The log contains 62.5 millions of query request records with the URLs of the corresponding clicked web pages during 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
Search engine	google.com	3452	yahoo.com	2179

We selected the top 10K popular queries in this query log and crawled all the corresponding clicked web pages (20K in total). 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 web pages they relate to.
- The similarity of two web pages can be reinforced by the similarity of the queries as well as the similarity of other web pages they relate to.

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:

$$L_{urm} = \begin{vmatrix} \alpha L_q & (1-\alpha)L_{qd} \\ (1-\alpha)L_{dq} & \alpha L_d \end{vmatrix} \quad (14)$$

where L_q is the query inter-type relationship matrix. Since there is no intra-type relationship in 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 hyper-link relationship during the similarity reinforcement process. Each sub-matrix in (14) is normalized to a row-stochastic matrix. A ***USM*** also can be created as:

$$S_{urm} = \begin{vmatrix} S_q & S_{qd} \\ S_{qd}^T & S_d \end{vmatrix} \quad (15)$$

where S_q is the query content similarity matrix, giving similarity between queries. S_d is the web page content similarity matrix; S_q and S_d are measured using *tf*idf* [28]. S_{qd} is the query web-page similarity matrix. Since it is not possible to measure the similarity between queries and web pages at the beginning, S_{qd} is set to 0. Then we apply the ***SimFusion*** algorithm on (14) and (15) to iteratively calculate the similarities of queries and web pages. The performance of the ***SimFusion*** algorithm will be compared with the pure content similarity measurement (e.g., *tf*idf*) and the ***SimRank*** algorithm.

5.2 Evaluation Metrics

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:

$$precision \quad at \quad N = \frac{\#\ of\ \ similiar\ \ objects}{N} \quad (16)$$

10 human experts were hired to manually identify the similar objects returned by different algorithms. The final judgment of relevancy was decided by majority vote.

5.3 Experimental Results

We set α=0.5 and developed the ***URM*** as in Eq. (14) and developed ***USM*** as in Eq. (15). Then, we iteratively calculate the ***SimFusion*** algorithm until convergence (9 iterations in our experiment). Randomly selected sets of queries and URLs are used to evaluate the effectiveness of the ***SimFusion*** algorithm. The results are reported below:

5.3.1 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 and evaluate the precision at top 10 queries returned by ***SimFuion***, ***SimRank*** and ***tf*idf*** algorithm for each of the 30 queries. Then, we compare the average precision at 10 for the three algorithms. The results are shown below:

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

The table shows that ***SimFusion*** algorithm achieves a 13.6% improvement over the ***SimRank*** algorithm and a 67% improvement over the ***tf*idf*** algorithm in terms of precision at 10. A query-by-query breakdown for the improvement of ***SimFusion*** over ***SimRank*** is presented in Figure 4. We will also make a case study by analyzing the top 10 similar queries returned for the query "pizza hut" by different algorithms as shown in Table 3.

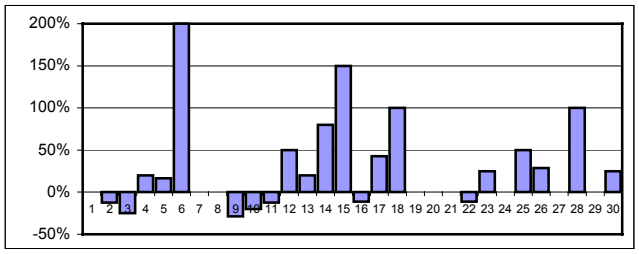

Figure 4: Query Breakdown for SimFusion vs. SimRank

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

Bold fond cells indicate similar queries. We can see that the ***tf*idf*** can not provide best results because it can only return content similar queries (e.g., pizza), on the other hand, ***SamRank*** can not achieve best performance either because it returns too many semantic "marginal" relevant queries (e.g, kfc, taco bell, burger king). ***SimFusion*** algorithm can be considered as a combination of the two extreme algorithms and can achieve best performance by returning both content similar (e.g., dominos pizza) and semantic similar (e.g., pap johns, dominos) queries in top results.

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-iteration curve in Figure 5 below:

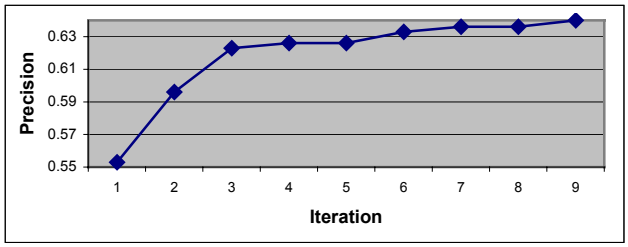

Figure 5: Precision vs. Iteration curve for SimFusion

We can see from Figure 5 that ***SimFusion*** improve the similarity measurement faster at the initial iterations than at the latter iterations. Similar findings had also been reported in [13].

5.3.2 Results on Similar Web pages

We use the similar evaluation metric used in 5.3.1 to evaluate the performance of ***SimFusion*** algorithm on web pages. We randomly chose 10 web pages from the log, for each of them we evaluate the similarity of the top 10 web pages returned by ***SimFusion*** and ***SimRank*** algorithms respectively. After evaluation we found that the average precision for ***SimFusion*** algorithm is 0.8, and is 16% better than the ***SimRank*** algorithm, which achieves an average precision of 0.69. A detailed precision comparison of the ***SimFusion*** and ***SimRank*** algorithm for the 10 web pages are shown in Figure 6 below.

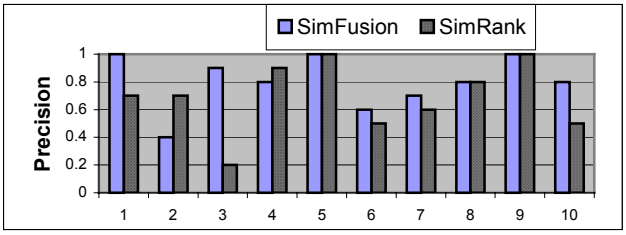

Figure 6: SimFusion vs. SimRank on web page similarities

We also conduct a case study by analyzing the top similar web of "http://www.imdb.com", returned by the two algorithms.

Table 4. Case Study for web page "www.imdb.com"

SimFusion	*SimRank*
http://movies.msn.com/default	**http://www.imdb.com**
http://www.imdb.com	**http://movies.go.com**
http://movies.go.com	**http://www.ifilm.com**
http://www.amazon.com/exec/obidos/tg/browse/-/130//	http://www.rame.net/faq/deadporn
http://video.barnesandnoble.com/home_cds2.asp?pid=1411&sourc	http://www.bbc.co.uk/learning/
http://www.movieclicks.com	http://www.bbc.co.uk/radio1
http://movies.channel.aol.com	**http://www.hollywood.com/index.html**
http://www.allmovie.com	http://www.absolutepictures.com/l/lords_traci
http://www.reel.com	http://www.bbc.co.uk/radio
http://www.mrqe.com	**http://www.reel.com**

Bold fond cells indicate similar queries in the table above. We can see that the reason the ***SimFuison*** algorithm out performances the ***SimRank*** algorithm is that ***SimFuion*** has returned more content similar web pages (e.g., www.movieclicks.com) than the ***SimRank*** algorithm while still keeps some semantic relevant web pages returned by the ***SimRank*** algorithm (e.g., www.reel.com).

5.3.3 Parameter selection

We also investigate how different values of α defined in Eq. (14) can affect the performance of the ***SimFusion*** algorithm. We chose 10 α values from 0 to 1 at the interval of 0.1 and evaluate the performance of ***SimFusion*** on finding similar queries at each α value and draw the performance curve below:

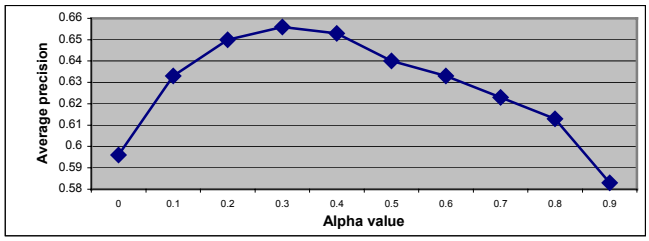

Figure 7: Performance curve for different α values in SimFusion algorithm

We can see from Figure 7 that the ***SimFusion*** algorithm achieve 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 calculated the similarity of different queries. When annotating the data, we also found that, different queries find their best similar queries at different α values, thus, it will be interested to think whether it is possible to automatically determine a set of parameters for each data object being considered in the ***URM***.

6. CONCLUSION AND FUTURE WORK

In this paper, we highlighted the research topic of web information integration, that is: *"*How can the broad variety of heterogeneous data and relationships in the web be effectively integrated to improve the performance of various information applications?*"* In order to better understand this, we introduced the Unified Relationship Matrix (***URM***) to represent heterogeneous data objects and their relationships in a unified manner. Then, we tackled the information integration problem by analyzing 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 demostrate 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 will use machine-learning technologies to automatically determine the values of the parameters in the ***URM***, and to optimize the performance of the ***SimFusion*** algorithm. We also will improve the efficiency of the ***SimFuion*** algorithm by pruning the ***URM*** as well as parallelizing the calculation, so that it can be easily applied to popular applications (i.e., the large scale mining application used in Google).

7. REFERENCES

[1] S. Acid, L. M. D. Campos, J. M. Fernandez-Luna, and J. F. Huete, "An Information Retrieval Model Based on Simple Bayesian Networks," *International Journal of Intelligent Systems*, vol. 18, pp. 251-265, 2003.

[2] 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.

[3] 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, 1971, Chapter 24.

[4] 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.

[5] P. Calado and B. Ribeiro-Neto, "An Information Retrieval Approach for Approximate Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, pp. 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] B. D. Davison, "Toward a unification of text and link analysis." *in 26th annual international ACM SIGIR conference on research and development in information retrieval,* Toronto, Canada, pp. 367-368. 2003.

[8] 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.

[9] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis." *Journal of the Society for Information Science, 41(6)*, pp. 391-407, 1987.

[10] S. T. Dumais, G. W. Furnas, T. K. Landauer, and S. Deerwester, "Using latent semantic analysis to improve information retrieval." In *Proceedings of CHI'88: Conference on Human Factors in Computing*, New York: ACM, pp. 281-285.

[11] E. Fox. "Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types". *Cornell University Dissertation*, Aug. 1983.

[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] 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.

[14] E. Ide. "New experiments in relevance feedback", in *The SMART Retrieval System*, G. Salton, editor, Prentice Hall, 1971, pp. 337-354.

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

[16] J.M. Kleinberg, Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46 (5), pp. 604-632.

[17] 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.

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

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

[20] J. Pitkow and P. Pirolli. "Life, death, and lawfulness on the electronic frontier". *In Proceedings of the Conference on Human Factors in Computing Systems*, Atlanta, Georgia, pp. 383-390, 1997.

[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] E. Rasmussen. Clustering algorithm. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structure and Algorithms*, Chap. 16. Prentice Hall, 1992.

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

[25] 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.

[26] 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.

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

[28] G. Salton and M. J. McGill. "Introduction to Modern Information Retrieval". McGraw-Hill Book Co., New York, 1983.

[29] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents., 24:265-269, 1973.

[30] Z. Su, Q. Yang, H.-J. Zhang, X. Xu, Y. Hu, "Correlation-based Document Clustering using Web Logs", *in Proceedings of the 34th Hawaii International Conference on System Science*, Hawaii, U.S.A. 2001.

[31] H. Turtle and W. B. Croft, "Inference networks for document retrieval," *In Proceedings of the13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Brussels, Belgium, 1990.

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

[33] J. D. Wang, H. J. Zeng, Z. Chen, H. J. Lu, L. Tao, and W.-Y Ma. "ReCoM: reinforcement clustering of multi-type interrelated data objects". In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada, pp. 274-281, July 2003.

[34] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong, "On Modeling of Information Retrieval Concepts in Vector Space," *ACM Transactions on Database Systems*, vol. 12, pp. 299-321, 1987.

[35] W. Xi, B. Zhang, Z. Chen, Y. Lu, S. Yan, W.Y. Ma, E.A. Fox. "Link Fusion: A Unified Link Analysis Framework for Multi-type Inter-related Data Objects", *in Proceedings of the 13th International World Wide Web Conference, WWW2004,* pp. 319-327, New York, U.S.A. May 19-22, 2004.

[36] G. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, W. Xi, Y. Yu, E.A. Fox,"MRSSA: An Iterative Algorithm for Similarity Spreading over Interrelated Objects*", in Proceedings of the 13th Conference on Information and Knowledge Management, Washington*, D.C. Nov. 8-13, 2004, to appear.

[37] G. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, W. Xi, W. Fan, Y. Yu "Optimizing Web Search Using Web Click-through Data"*, in Proceedings of the 13th Conference on Information and Knowledge Management,* Washington D.C., U.S.A, Nov. 8-13, 2004, to appear.

APPENDIX

Proof of Convergence for the SimFusion algorithm

We prove the convergence of iterative equation (11). The proof of convergence for Eq. (10) is similar to that of Eq. (11) if we have $\alpha = 0$ or $\alpha = 1$. To prove Eq. (11), two definitions are needed first.

Definition 1: Given matrices $A \in R^{m\times n}$, $B \in R^{p\times q}$, then their *Kronecker Product* $A \otimes B$ is,

$$A\otimes B=\begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}_{mp\times nq}$$

Definition 2: the *Row-First Vectorization* of a matrix $A \in R^{m\times n}$, denoted as $\vec{A}$, could be represented as $\vec{A} = (a_1, a_2, \cdots, a_m)^T$, where $a_i \in R^n$, $i = 1, 2, \cdots, m$ are row vectors of A.

Lemma 1: for matrices $A \in R^{m\times n}$, $B \in R^{n\times n}$, the *Line-First Vectorization* of matrix ABA^T is equal to a vector $(A\otimes A)\vec{B} \in R^{m^2}$.

Proof: from definition 1 and definition 2,

$$(A\otimes A)\vec{B} = \begin{pmatrix} a_{11}A & a_{12}A & \cdots & a_{1n}A \\ a_{21}A & a_{22}A & \cdots & a_{2n}A \\ \vdots & \vdots & & \vdots \\ a_{m1}A & a_{m2}A & \cdots & a_{mn}A \end{pmatrix}\begin{pmatrix} b_{11} \\ b_{12} \\ \vdots \\ b_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^{m}\sum_{j=1}^{n} a_{1,i}a_{1,j}b_{i,j} \\ \sum_{i=1}^{m}\sum_{j=1}^{n} a_{1,i}a_{2,j}b_{i,j} \\ \vdots \\ \sum_{i=1}^{m}\sum_{j=1}^{n} a_{m,i}a_{m,j}b_{i,j} \end{pmatrix}_{mm\times 1} = \vec{(ABA^T)}$$

Then using lemma 1, the iterative Eq. (11), can be transformed into $\vec{S}^{n}_{usm} = \alpha(L^{out}_{urm} \otimes L^{out}_{urm})\vec{S}^{n-1}_{usm} + (1-\alpha)(L^{in}_{urm} \otimes L^{in}_{urm})\vec{S}^{n-1}_{usm}$

$= (\alpha(L^{out}_{urm} \otimes L^{out}_{urm}) + (1-\alpha)(L^{in}_{urm} \otimes L^{in}_{urm}))\vec{S}^{n-1}_{usm}$

Let matrix $H = \alpha(L^{out}_{urm} \otimes L^{out}_{urm}) + (1-\alpha)(L^{in}_{urm} \otimes L^{in}_{urm})$, then the iterative equation (11) can be rewritten as: $\vec{S}^{n}_{usm} = H\vec{S}^{n-1}_{usm}$. ■

Lemma 2: The matrix $H = \alpha(L^{out}_{urm} \otimes L^{out}_{urm}) + (1-\alpha)(L^{in}_{urm} \otimes L^{in}_{urm})$ is a non-negative, row-stochastic matrix.

Proof: Step1, H is a non-negative matrix:

Without loss of generality, both L^{in}_{urm} and L^{out}_{urm} can be denoted by (7). It is obvious that (7) is a non-negative matrix since L_{ij} are non-negative sub-matrices and $\lambda_{ij} > 0,\ 1 \le i, j \le N$. From the definition of *Kronecker Product* (definition 1), we know that $L_{urm} \otimes L_{urm}$ should be a non-negative matrix. In other words, both $L^{in}_{urm} \otimes L^{in}_{urm}$ and $L^{out}_{urm} \otimes L^{out}_{urm}$ are non-negative matrices. Since $0 \le \alpha \le 1$, H must be a non-negative matrix.

Step2, H is a row-stochastic matrix:

From its definition in Section 4.2 we know that matrix (7) is a row-stochastic matrix. Since the Kronecker Product of two row-stochastic matrices is still a row-stochastic matrix, then $L_{urm} \otimes L_{urm}$ should be a row-stochastic matrix. In other words, both $L^{in}_{urm} \otimes L^{in}_{urm}$ and $L^{out}_{urm} \otimes L^{out}_{urm}$ are row-stochastic matrices. Moreover, since $0 \le \alpha \le 1$, H must be a row-stochastic matrix. ■

Lemma 3: If L_{urm} is non-negative, row-stochastic and reducible, there exists a permutation matrix P, such that $PHP^T = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$. Here, H_1 is a non-negative, row-stochastic, and irreducible matrix.

Proof: Note that from the definition of Kronecker Product, $L_{urm} \otimes L_{urm}$ shall preserve the symmetry property of L_{urm}. Then the proof of lemma 3 is the same as the proof of lemma B in the appendix of [35], if we use L_{urm} to replace L'_{urm}.■

Theorem 1: For the unified matrices L^{out}_{urm} and L^{in}_{urm} defined in Eq. (11), iterative method $\vec{S}^{n}_{usm} = H\vec{S}^{n-1}_{usm}$ converges to the principle eigenvector of $H = \alpha(L^{out}_{urm} \otimes L^{out}_{urm}) + (1-\alpha)(L^{in}_{urm} \otimes L^{in}_{urm})$.

From lemma 2 and lemma 3, and lemma C in the Appendix of [35], we know that if H is irreducible, the iterative method $\vec{S}^{n}_{usm} = H\vec{S}^{n-1}_{usm}$ converges to the principle eigenvector of H. Similar to the proof in appendix of [35], if we use H to replace L'_{urm}, and use $\vec{S}^{n}_{usm} = H\vec{S}^{n-1}_{usm}$ to replace $w = L'^{T}_{urm} w$, then the theorem in the appendix of [35] tells us that iterative method $\vec{S}^{n}_{usm} = H\vec{S}^{n-1}_{usm}$ converges to the principle eigenvector of H. ■