SHINE: Search Heterogeneous Interrelated Entities

Xuanhui Wang
University of Illinois at Urbana-Champaign
Urbana, IL 61801
xwang20@cs.uiuc.edu

Jian-Tao Sun
Microsoft Research Asia
Haidian District
Beijing, China 100080
jtsun@microsoft.com

Zheng Chen
Microsoft Research Asia
Haidian District
Beijing, China 100080
zhengc@microsoft.com

ABSTRACT

Heterogeneous entities or objects are very common and are usually interrelated with each other in many scenarios. For example, typical Web search activities involve multiple types of interrelated entities such as end users, Web pages, and search queries. In this paper, we define and study a novel problem: Search Heterogeneous INterrelated Entities (SHINE). Given a SHINE-query which can be *any type(s)* of entities, the task of SHINE is to retrieve multiple types of related entities to answer this query. This is in contrast to the traditional search, which only deals with *a single* type of entities (e.g., Web pages). The advantages of SHINE include: (1) It is feasible for end users to specify their information need along different dimensions by accepting queries with different types. (2) Answering a query by multiple types of entities provides informative context for users to better understand the search results and facilitate their information exploration. (3) Multiple relations among heterogeneous entities can be utilized to improve the ranking of any particular type of entities. To attain the goal of SHINE, we propose to represent all entities in a unified space through utilizing their interaction relationships. Two approaches, M-LSA and E-VSM, are discussed and compared in this paper. The experiments on 3 data sets (i.e., a literature data set, a search engine log data set, and a recommendation data set) show the effectiveness and flexibility of our proposed methods.

Categories and Subject Descriptors: H.3.1 [Content Analysis and Indexing]: Indexing methods; H.3.3 [Information Search and Retrieval]: Search process

General Terms: Algorithms, Design

Keywords: SHINE, heterogeneous interrelated entities, multiple types, search

1. INTRODUCTION

Heterogeneous interrelated entities or objects are common in many scenarios. For example, general Web search activities involve heterogeneous objects including end users, search queries, Web pages, and words. In literature data, four types of heterogeneous entities are included: authors, papers, conferences, and keywords. All the inter-relationships among these heterogeneous objects provide rich information to build potentially better search services. However, most of current search services only deal with a single type of objects (e.g., Web pages) and are limited in both query capabilities and search result richness. For example, in the traditional Web search, end users can only specify their information need by a string of query terms and only a ranked list of pages are returned as search results. Although most search engine users have been trained to become accustomed to this traditional method, several drawbacks are still needed to be addressed in order to better satisfy users' information needs. For example, in some cases, a user who needs help to formulate a better query may want to know "what are the possible *words* for me to refine my queries" or "what are the related *queries* other users submitted for similar needs?" In other cases, a Web master/advertiser is more curious about "what are the related *pages* to my *page*" or "what are the *queries* which are submitted by end users and related to my *page*?" For a literature search, a researcher may want to find *papers* of particular topics published in particular *conferences* or published by particular *authors*. A novice may want to know the prestigious *conferences* or *experts* of a research topic such as "Web search". In all these situations, we see keen needs from a user's perspective to extend both query capabilities and search result richness.

Fortunately, the rich information among heterogeneous interrelated objects provide feasibility to satisfy users' diverse information needs. In Figure 1, we show the interactions among heterogeneous entities: users, queries, Web pages, and words as following: users interact with queries by *issuing*; queries interact with Web pages by *referencing*; Web pages interact with words by *containing*; and so on. By leveraging all the interactions among heterogeneous entities, in this paper, we define and study a novel search problem: Search Heterogeneous INterrelated Entities (SHINE) to extend and generalize traditional searches. Specifically, to extend query capabilities, a SHINE-query can be any combinations of entities in multiple types. To extend search result richness, SHINE retrieves and returns all types of heterogeneous entities. For example, given a SHINE-query such as "xbox" with the type of *word*, the search results of SHINE consist of relevant users, Web pages, text-queries, and related words. On the other hand, a user can input a Web page such as `www.xbox.com` as a SHINE-query to get relevant results consisting of all the four types of entities.

The functionalities of SHINE make such search services more desirable. Take literature search service as an example:

- The returned heterogeneous entities provide informative context for users to understand the results. For example, a SHINE-query "data mining" with the type of *word* will return the relevant authors such as "Jiawei

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

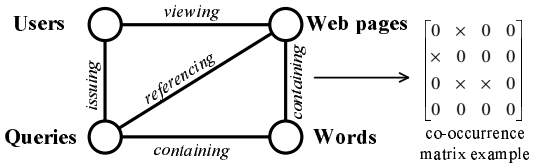

Figure 1: Example of Heterogeneous Interrelated Entities. Each edge denotes a single interaction relationship, which corresponds to a co-occurrence matrix.

Han" (`http://www-sal.cs.uiuc.edu/~hanj`), conferences like "KDD" (`http://www.acm.org/sigs/sigkdd`), words like "pattern" and "association", etc. A novice who is interested in "data mining" can not only find relevant papers, but also active researchers/professors, prestigious international conferences, and semantically related keywords. All these contextual information is very useful to help the user be familiar with "data mining" field.

- The query capabilities of SHINE can facilitate users to specify the desired information along multiple dimensions. Given a SHINE-query, SHINE will return relevant entities in different types. If a user is interested in a returned entity and wants to know more details, she can input this entity or its combination with others as a new SHINE-query to get refined results. For example, a user may first submit a SHINE-query with the type of word. After reading the returned results, she may submit her second query with the type of author. This provides more flexibility for users' information exploration.

- In contrast to the traditional search which only considers a single interaction relationship, SHINE can utilize all these complementary interaction relationships to help retrieve semantically related entities more reliably. For example, two queries in Figure 1 can be similar not only because they contain similar words, but also because they refer to similar Web pages or are issued by similar users.

However, the task of SHINE is challenging in the following aspects. (1) Different from traditional search problems which only deal with a single type of entities, the goal of SHINE is to search heterogeneous entities which are of *multiple* types. It is not clear how to relate the *same* query to *different types* of entities. (2) There exist hidden semantics underneath all interactions among heterogeneous entities. Identifying the latent salient concepts is important to find semantically related entities. (3) In real applications, the entity number may be huge. As more and more entities emerge (e.g., entities in search log), the number of entities may increase dramatically. Thus the scalability becomes a big issue and an efficient solution to SHINE is more desirable.

To the best of our knowledge, this problem has not been well studied in previous literature. In this paper, we first formally define the SHINE problem and then we propose a unified framework to address it. Two approaches are discussed within this framework: Multiple-type Latent Semantic Analysis (M-LSA) [23] and Extended Vector Space Model (E-VSM). The first approach M-LSA is a generalization of traditional Latent Semantic Analysis (LSA) [12]. Its advantages include: (1) It can represent all heterogeneous objects in a unified space. (2) It conducts latent semantic analysis and identifies latent salient concepts underneath all heterogeneous interaction relationships. The drawback of M-LSA is its inefficiency. Motivated by the unified representation of M-LSA, the second approach E-VSM is proposed as a very efficient solution to SHINE. E-VSM is an extension of the traditional Vector Space Model (VSM) and is shown to be linearly scalable in off-line indexing and sublinear in on-line searching. With the unified representation of heterogeneous objects in E-VSM, important information retrieval techniques, *ranking* and *feedback*, can be incorporated naturally. As shown in [3], the essence of spectral methods is to conduct "document expansion". Feedback in our model is to conduct "query expansion", which could achieve similar effects as document expansion. Thus combining feedback with E-VSM can help find semantically related objects.

To test the effectiveness of our proposed methods, we conduct experiments on 3 data sets including a literature data set, a search engine log data set, and a recommendation data set. The experiments show that our SHINE formulation is very flexible and effective for different types of tasks.

The rest of this paper is organized as follows. We review the related work in Section 2. Section 3 is to formally define SHINE problem and we describe our unified framework in Section 4. Our system is described in Section 5. We adapt this system into 3 different tasks and conduct experiments in Section 6. Finally, we conclude this paper and discuss future work in Section 7.

2. RELATED WORK

Heterogeneous interrelated entities have attracted lots of attentions recently. For example, several recent work studies the effectiveness of clustering different types of objects by utilizing the interaction information [25, 22, 16, 4]. They find that the clustering results can be improved compared with the results when they only consider a single type of interaction relationship. Other work such as SimRank [14] measures the similarity between different objects by utilizing the interactions among heterogeneous objects. [23] proposes to conduct latent semantic analysis on these heterogeneous interrelated objects, which is a generalization of the work [11]. In this paper, we utilize the heterogeneous relationships among different types of entities to improve users' search experiences.

PageRank [6, 18] and HITS [15] are the two earliest link analysis algorithms. They only consider a single type of relationship (i.e., hyper-links) among homogeneous Web pages. Recent work such as [26, 17, 1] extends PageRank or HITS to consider the interactions among heterogeneous entities. All these algorithms focus on improving the estimation of entities' static ranking, which is query independent. In contrast, we are studying how to extend query capabilities and search result richness by utilizing the relationships among heterogeneous entities.

Instead of solely returning Web pages, several recent work proposes to answer search queries by a certain type of entities in a finer granularity [8, 9, 10]. For example, in [8], "object finder" queries are defined to find the top K objects that match a given set of keywords. Their algorithm first retrieves relevant documents and then relevant objects

are ranked according to their relationship with the returned documents. "Expert search" track was initiated by TREC conference[1] recently and its task is to retrieve experts given a topic description. Our work is more general since all these work only considers retrieving a specific type of objects and the query can only be a set of keywords.

SHINE is also related to several commercial systems such as Google Scholar[2] and Citeseer[3], when we apply our technologies to the literature domain. A recent feature of Google Scholar is that it also provides an author list besides retrieved papers given a query. This is a similar feature to SHINE in which authors are used to answer an input query. However, there is no public research about how they rank the authors. Furthermore, we are studying a more general problem which can also return other objects (e.g., conferences) and can be applied to data sets in other domains (e.g., search engine logs).

3. THE SHINE PROBLEM

In this section, we formally define our research problem: Search Heterogeneous INterrelated Entities (SHINE). We first describe and model the data formally. Then we define the SHINE problem.

3.1 Heterogeneous Interrelated Entities

Suppose we have N types of objects $\{X_1, X_2, ..., X_N\}$ and each pair of them could have an interaction relationship. We model the objects and their interactions using a graph as defined in the following:

DEFINITION 1 (MULTIPLE-TYPE GRAPH). *A multiple-type graph $G(V, E)$ consists of N vertices with the i-th vertex corresponding to the i-th type of entities X_i. If two types of entities X_i and X_j have an interaction relationship, there is one edge $e_{ij} \in E$ connecting the i-th and j-th vertices.*

For example, in Figure 1, the corresponding graph G contains 4 types of objects: users, queries, Web pages, and words. We have 5 interaction relations in G and each of them is denoted by an edge in Figure 1. In general, all the interaction relationships can be represented as co-occurrence matrices with each entry measuring the correlation strength between two corresponding entities. For example, in information retrieval using the "bag-of-words" method, the interaction between words and documents is represented by a co-occurrence matrix with each entry measuring the word importance in the corresponding document [2]. In this way, each edge e_{ij} in a multiple-type graph corresponds to a $|X_i| \times |X_j|$ matrix M_{ij}. In Figure 1, the 5 co-occurrence relationships correspond to 5 matrices and an example is also included in Figure 1. For a particular application, different matrices may have different importance and we can associate a weight α_{ij} with edge e_{ij} to reflect its relative importance.

The multiple-type graph encodes the semantics of all the interactions among heterogeneous entities. Semantically related objects may directly co-occur with each other or may co-occur via other types of objects. Take the literature search as an example, two researchers may be related because they have co-authored papers. They can also be implicitly related because their papers are published in related conferences or their papers are on the same topics. Our task of SHINE is to exploit these interactions on a multiple-type graph to answer a query by semantically related data objects.

[1]http://trec.nist.gov/
[2]http://scholar.google.com
[3]http://citeseer.ist.psu.edu

3.2 SHINE Formulation

On a multiple-type graph, the task of SHINE is to retrieve all types of relevant objects given a query. In this section, we give the definition of a SHINE-query and the expected results in SHINE framework.

DEFINITION 2 (SHINE-QUERY). *A query Q in SHINE framework is an object of any type or a list of objects of multiple types. In general, $Q := [t_i : ID_i]_{i=1..k}$, where k is the number of objects in query Q and $[t_i : ID_i]$ means the type of the i-th object in Q is t_i and its identifier is ID_i.*

For example, in literature search task, the query $Q := [author : Jiawei\ Han; word : mining]$ is a combination of two objects. The first object is an *author* and its identifier is *Jiawei Han* and the second is a *word* and its identifier is *mining*. Just as the "bag-of-words" method used in ad-hoc information retrieval, we can treat a SHINE-query Q as "bag-of-objects".

DEFINITION 3 (SHINE RESULTS). *Given a SHINE-query Q and a multiple-type graph G, the SHINE search results are several lists of relevant objects. All the objects in a list are of the same type and ranked by its relevance with respect to the query Q.*

Table 1 presents a mock up result example of the given query $Q := [author : Jiawei\ Han; word : mining]$ in the literature search task. Here we have 4 types of objects thus we have 4 lists. Each list is ranked according to the relevance of the objects with respect to the query Q. For example, the first conference is "KDD" and the first keyword is "pattern", which are the most relevant conference and the most relevant keyword to the query Q respectively.

papers	authors	conferences	keywords
p1	Philip_Yu	KDD	pattern
p2	Xifeng_Yan	SIGMOD	association
⋮	⋮	⋮	⋮

Table 1: An example of the SHINE search results in the literature search application.

As we can see, the advantages of SHINE include: (1) SHINE provides more flexibility for users to specify the desired information by SHINE-queries which are not restricted to several keywords. We can add any type of objects as components into SHINE-queries. For example, if a user only wants to know the "mining" papers published by "Jiawei Han", she can compose a query $Q := [author : Jiawei\ Han; word : mining]$. (2) SHINE returns different types of entities which enrich the information of the search results and provide useful context for users to digest the information. This can also help to answer interesting questions such as: "Who are the active researchers in SIGIR conference?" "What keywords can be used to annotate a researcher or a conference such as CIKM?" (3) SHINE search results also facilitate users' information exploration. For example, a user may refine her query from

$Q := [author : Jiawei\ Han; word : mining]$ to $Q := [word : mining; conference : KDD]$ if she wants to know more along the conference dimension.

In the experiment part, we will build SHINE search services on a literature data set and a commercial search engine log data set. We will show that more interesting questions or information needs can be satisfied by our SHINE functionalities.

4. OUR SOLUTION TO SHINE

In this section, we propose a unified framework for searching heterogeneous objects. Within this framework, all heterogeneous objects are represented in a unified vector space. Thus the relevance between any two objects (even in different types) can be measured in this space and the ranking and feedback techniques used in the traditional information retrieval can be incorporated in our framework naturally.

4.1 Object Representations

We first describe two approaches to represent heterogeneous objects by utilizing the interactions among them: Multi-type Latent Semantic Analysis (M-LSA) based method and Extended Vector Space Model (E-VSM) based method.

4.1.1 M-LSA based Method

In our previous work [23], M-LSA is proposed to conduct latent semantic analysis using interactions among heterogeneous objects. Given a multiple-type graph G, M-LSA is used to identify the most important concepts contained in the co-occurrence data. As a result, all heterogeneous objects are mapped into a low-dimensional semantic space. The importance of concepts are identified by the Mutual Reinforcement Principle (MRP):

> On a multiple-type graph G with a number of vertices and pairwise co-occurrence relationships, *important* objects of a type co-occur with *important* objects of other types.

Formally, assume we have N types of objects on graph G: $\{X_1, X_2, ..., X_N\}$. For any two types of objects: X_i and X_j, we have the co-occurrence matrix M_{ij} ($M_{ij} = 0$ if the edge e_{ij} is absent on G). Let us associate an importance value with each object. For the i-th type of objects in X_i, we have one weight vector $\mathbf{w}_i$ to denote their importance. The mutual reinforcement principle can be expressed as:

$$\mathbf{w}_i \doteq \sum_{\forall j: j \neq i} \alpha_{ij} M_{ij} \mathbf{w}_j \tag{1}$$

where α_{ij} reflects the relative importance of matrix M_{ij}.

Taking a unified view of the latent concepts, we use $\mathbf{w} = [\mathbf{w}_1, ..., \mathbf{w}_N]^T$ as the concatenated importance vector and define

$$R = \begin{bmatrix} 0 & \alpha_{12}M_{12} & \cdots & \alpha_{1N}M_{1N} \\ \alpha_{21}M_{21} & 0 & \cdots & \alpha_{2N}M_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N1}M_{N1} & \alpha_{N2}M_{N2} & \cdots & 0 \end{bmatrix} \tag{2}$$

as the unified co-occurrence matrix. We can rewrite Equation (1) in a matrix format:

$$\mathbf{w} \doteq R \cdot \mathbf{w} \tag{3}$$

It is easy to know that $\mathbf{w}$ will converge to the eigenvector of the co-occurrence matrix R.

In M-LSA, each $\mathbf{w}$ is regarded as a latent concept underneath all the co-occurrence relations. Each entry in $\mathbf{w}$ corresponds to an object and its value can be regarded as the association weight between the object and this latent concept. Similarly, the first k eigenvectors of R represent the top k most important concepts, which span a k-dimensional semantic space to represent all the objects. Specifically, let

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k$$

be the top k eigenvalues of R and the corresponding vectors are

$$\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k.$$

λ_l gives precisely the salience of the corresponding concept vector $\mathbf{c}_l$ ($1 \leq l \leq k$). Therefore, the i-th object can be represented by

$$[\lambda_1 c_{1i}, \lambda_2 c_{2i}, ..., \lambda_k c_{ki}]$$

where c_{li} is the i-th entry in $\mathbf{c}_l$ (i.e., the association weight between the i-th object and the l-th concept). All the objects can be represented in a matrix:

$$[\lambda_1 \cdot \mathbf{c}_1, \lambda_2 \cdot \mathbf{c}_2, ..., \lambda_k \cdot \mathbf{c}_k]$$

with each row representing an object in the k-dimensional space.

M-LSA based method has the advantage of utilizing all the co-occurrence relations to capture the latent semantics on a multiple-type graph. Unfortunately, its computational cost is expensive because it involves solving an eigenvector problem of a matrix with dimension equal to the total number of objects. This makes it difficult to apply M-LSA to large scale data sets.

4.1.2 Extended Vector Space Model

Motivated by the unified representation of M-LSA, we propose an efficient method: Extended Vector Space Model (E-VSM). E-VSM is closely related to M-LSA in that it represents all heterogeneous objects in a unified space.

E-VSM is an extension of traditional vector space model. In traditional vector space model, given a term by document co-occurrence matrix $A = [a_{ij}]$, each document is represented in a word space, which corresponds to a column vector in A, and each word is represented in a document space, which corresponds with a row vector in A. To seek a unified space to represent both documents and words, we "concatenate" the two spaces spanned by words and documents and represent each object (document or word) by a unified longer vector. By filling zeroes in the missing dimensions of each object, we get a unified matrix $\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$. However, this type of representations do not help since a word and a document have no overlapped dimensions with nonzero values, and thus their similarity score is still zero in this space. Clearly, we can replace the two 0 matrices by the co-occurrence information among documents and among words respectively. In the worst case where this co-occurrence information is not available, we can still assume that an object co-occurs with itself by default, and thus replace 0 matrices by the identity matrix I. We get:

$$U = \begin{bmatrix} I & A \\ A^T & I \end{bmatrix} \tag{4}$$

Each row in the upper part is a document representation vector and each row in the lower part is a word representation vector.

On a multiple-type graph, each type of objects co-occur with other types of objects. Similar to the document and word representations, an object can be represented by several vectors. Each vector corresponds to a type of objects which co-occur with the considered object. Formally, given a multiple-type graph G with N vertices, the i-th type of objects can be represented by the j-th type of objects via the co-occurrence matrix M_{ij}. By concatenating all the matrix together, we can represent each object by a unified longer vector. We again can assume that each object co-occurs with itself by default. Then we obtain a new matrix U:

$$U = \begin{bmatrix} I & M_{12} & \cdots & M_{1N} \\ M_{21} & I & \cdots & M_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ M_{N1} & M_{N2} & \cdots & I \end{bmatrix} \quad (5)$$

In the unified matrix U, each object corresponds to a row vector. All the objects are represented in a unified space spanned by all the objects on the multiple-type graph G. Similar to M-LSA, we can associate an importance value α_{ij} with each matrix M_{ij} and thus transform U to a weighted matrix

$$U = \begin{bmatrix} I & \alpha_{12}M_{12} & \cdots & \alpha_{1N}M_{1N} \\ \alpha_{21}M_{21} & I & \cdots & \alpha_{2N}M_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N1}M_{N1} & \alpha_{N2}M_{N2} & \cdots & I \end{bmatrix} \quad (6)$$

E-VSM is closely related to M-LSA. In fact, the following Theorem 1 describes their relation.

THEOREM 1. *Matrix U in Equation (6) and matrix R in Equation (2) have the same eigenvectors.*

PROOF. It is easy to show that $U = I + R$. Suppose $\mathbf{c}_i$ is an eigenvector of R with eigenvalue λ_i, we have $R \cdot \mathbf{c}_i = \lambda_i \cdot \mathbf{c}_i$. Thus $U \cdot \mathbf{c}_i = \mathbf{c}_i + R \cdot \mathbf{c}_i = (1 + \lambda_i) \cdot \mathbf{c}_i$. This shows that $\mathbf{c}_i$ is an eigenvector of U with eigenvalue $1 + \lambda_i$. Similarly, it can be shown that eigenvectors of U are also eigenvectors of R. □

Both M-LSA and E-VSM represent all the objects in a unified space. From Theorem 1, we know that M-LSA is the representation of E-VSM in a latent semantic space, and thus a potentially better representation. There are several advantages in a unified space: (1) Since all the objects are represented in the same space, we can calculate the similarity between heterogeneous objects. (2) This representation considers multiple and complementary information on G and thus can measure the similarity between objects more accurately. (3) This can facilitate to incorporate the useful techniques such as ranking and feedback used in the traditional information retrieval into SHINE and we will discuss this in the next section.

4.2 Ranking and Feedback

One of the most important techniques in information retrieval is ranking [21]. Given a text-query consists of several words, documents are ranked according to the similarity scores between the documents and the query. We define the similarity measure for the unified representations in this section. Later we will show that the traditional VSM is a special case in our definition.

Since each object is represented in a unified space, a straightforward way to define the similarity is the inner product or cosine score of two vectors. In the following, we will only use inner product as the similarity measure since cosine similarity can be defined similarly. Specifically, the similarity between two object vectors $\mathbf{o}_1$ and $\mathbf{o}_2$ is:

$$S(\mathbf{o}_1, \mathbf{o}_2) = \sum_l o_{1,l} \cdot o_{2,l} \quad (7)$$

where $o_{1,l}$ ($o_{2,l}$) is the l-th value in $\mathbf{o}_1$ ($\mathbf{o}_2$).

Given a query $Q := [t_j : ID_j]_{j=1..k}$ in SHINE framework and let $\mathbf{o}_j$ be the object vector corresponding to the j-th object in Q, the similarity between Q and an object vector $\mathbf{o}$ is calculated as:

$$Sim(Q, \mathbf{o}) = \sum_{j=1}^{k} S(\mathbf{o}_j, \mathbf{o}) \quad (8)$$

It is easy to show that $Sim(Q, \mathbf{o}) = S(\mathbf{q}, \mathbf{o})$ where

$$\mathbf{q} = \sum_{j=1}^{k} \mathbf{o}_j. \quad (9)$$

Thus we can regard $\mathbf{q}$ as the vector representation of Q in the unified space.

All the objects can then be ranked according to their similarity defined in Equation (8). Since each object bears a type, we can separate the ranked list according to their types and thus obtain the SHINE results as defined in Definition 3.

An important and effective technique to improving the retrieval performance is feedback. We show that our object representation can incorporate feedback to conduct query expansion naturally. As discussed in [3], the essence of spectral methods like LSA is to conduct "document expansion" implicitly. In our E-VSM, feedback is used to conduct "query expansion" and thus could achieve similar effects as document expansion. Therefore, combining feedback with E-VSM can help find semantically related objects. Specifically, suppose that we have a set of relevant objects, C, which are either judged by users or the top retrieved results as in pseudo feedback [7], the expanded query vector $\tilde{\mathbf{q}}$ of $\mathbf{q}$ is calculated similarly to the Rocchio method [19]:

$$\tilde{\mathbf{q}} = \beta \cdot \mathbf{q} + \gamma \cdot \frac{1}{|C|} \sum_{\mathbf{o} \in C} \mathbf{o} \quad (10)$$

where β and γ are the weighting factors. By considering the different types of objects in C, the expansion can be improved as:

$$\tilde{\mathbf{q}} = \beta \cdot \mathbf{q} + \gamma \cdot \sum_{i=1}^{N} \frac{\alpha_i}{|C_i|} \sum_{\mathbf{o} \in C_i} \mathbf{o} \quad (11)$$

where C_i is the i-th type of feedback objects and α_i is its feedback weight. We add a constraint $\sum_{i=1}^{N} \alpha_i = 1$. Since $\tilde{\mathbf{q}}$ is still in the unified space, we can use the expanded query to retrieve and rank all the objects again to get refined results.

We have incorporated the ranking and feedback techniques into our proposed method. Both are extended from the traditional information retrieval techniques naturally since we represent all types of objects in a unified space.

4.3 Relation with Vector Space Model

In the traditional VSM, there are two types of objects involved: words and documents. In E-VSM, the unified matrix is the same as the matrix U in Equation (4). Therefore, the i-th word has a vector $\mathbf{w}_i$ and the j-th document has a vector $\mathbf{d}_j$ in the unified space. Suppose we have m words and n documents, then

$$\begin{aligned} d_{j,l} &= \begin{cases} 0 & 1 \le l \le n, l \ne j \\ 1 & l = j \end{cases} \\ w_{i,l} &= \begin{cases} 0 & n+1 \le l \le n+m, l \ne n+i \\ 1 & l = n+i \end{cases} \end{aligned} \quad (12)$$

Given a query Q which have k words, let $\mathbf{w}_{i(r)}$ be the vector corresponding to the r-th word in the query. Then, according to Equation (8), we have

$$\begin{aligned} Sim(Q, \mathbf{d}_j) &= \sum_{r=1}^{k} S(\mathbf{w}_{i(r)}, \mathbf{d}_j) \\ &= \sum_{r=1}^{k} \sum_{l=1}^{m+n} w_{i(r),l} \cdot d_{j,l} \\ &= \sum_{r=1}^{k} (\sum_{l=1}^{n} w_{i(r),l} \cdot d_{j,l} + \\ &\quad \sum_{l=n+1}^{n+m} w_{i(r),l} \cdot d_{j,l}) \\ &= \sum_{r=1}^{k} w_{i(r),j} + d_{j,i(r)+n} \end{aligned}$$

The last step is based on Equation (12). Since U is symmetric in Equation (4), we have $d_{j,i(r)+n} = w_{i(r),j}$, which is the co-occurrence frequency of $i(r)$-th word in j-th document. We have

$$\begin{aligned} Sim(Q, \mathbf{d}_j) &\propto \sum_{r=1}^{k} a_{i(r),j} \\ &= \sum_{t \in Q} c(t, Q) \cdot a_{t,j} \end{aligned}$$

Thus it is the dot product between the query vector and document vector in a space spanned by words, which is the traditional similarity score used in vector space model. Therefore, the traditional vector space model is a special case of E-VSM. If we have co-occurrence information within documents or co-occurrence information within words, we can incorporate this information into our framework straightforwardly. For example, the hyperlinks between Web pages can be regarded as co-occurrence information among Web pages and can be incorporated into our method easily.

5. SYSTEM OVERVIEW

We develop a general architecture for SHINE. Different applications can be adapted with little modification. Figure 2 gives the sketch of our system. It basically involves two parts: offline indexing and online searching. In the offline indexing part, the first step is to extract different co-occurrence relationships from the raw sources such as search engine logs. After we get each co-occurrence data, the next step is to build the unified indexing and push the the data to a repository. This step is the key step for the offline indexing part. We first build a unified dictionary to map each object string to a unique identifier. Then we go over this dictionary and build the representation of each object by iterating over all its co-occurred objects. We keep the type identity by assigning each object a type indicator. After this, we index all the objects and store the index into the repository. For the online searching part, the search interface accepts users' input, composes the input as a bag of objects, and sends it to the retrieval and ranking component. The retrieval component ranks all the objects according to their similarities to the SHINE-query. Since each object has a type indicator, the ranking component then separates the returned objects into different ranking lists and returns them to end users.

An snapshot of our system interface on literature data is given in Figure 3. In this figure, we have several input boxes where users can input objects of different types as queries. For example, the users can input a keyword, a conference, or an author, as a SHINE-query. The returned results include papers, authors, conferences, and keywords. Each type of returned objects are ordered by their relevance to the SHINE-query. To facilitate users' information exploration, we embed a hyperlink under each object descriptor. If a user clicks on a hyperlink, the system will automatically take the corresponding object as a new SHINE-query and return its corresponding results.

6. EXPERIMENTS

In this section, we conduct experiments to show the flexibility and effectiveness of our SHINE framework. We use three different data sets for experiments: a literature data set, a commercial search engine log data set, and a recommendation data set. We use E-VSM model as our solution to SHINE since these data sets are large.

6.1 SHINE Search on Literature Data

Organizing and searching literatures motivate the development of digital libraries such as ACM and IEEE Digital Libraries[4] and search engines such as Citerseer and Google Scholar[5]. In literature domain, researchers always want to find relevant and authoritative publications. In general, several different methods are used, including typing some keywords to a literature search engine, going to a well-known researcher's publication page, or going to a conference program page, to find relevant publications. Indeed, researchers implicitly try to find their paper objects using other objects such as keywords, authors, and conferences, as queries. In this section, we show that we can integrate all these different "methods" in our unified SHINE framework.

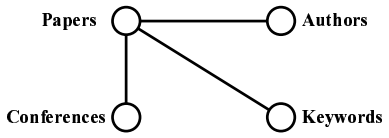

Figure 4: The multiple-type graph of literature data

6.1.1 Data Set

The data set we use in this experiment is the DBLP data set[6], which has information of all papers published by major computer science conferences. In this data set, each paper has an entry including its title, authors, published conference, and published year, etc. We define 4 types of objects for this data set: papers, authors, conferences, and keywords. The corresponding multiple-type graph is shown in Figure 4. In this data set, we have 463,931 papers, 350,538 authors, 2,957 conferences, 78,349 keywords, and thus 895,701 objects in total. Based on the multiple-type graph, we represent each object in a unified space by E-VSM model using Equation (6). We set all α_{ij}'s to 1 in this experiment.

[4]http://www.acm.org, http://www.ieee.org
[5]http://citeseer.ist.psu.edu, http://scholar.google.com
[6]available from http://dblp.uni-trier.de/xml

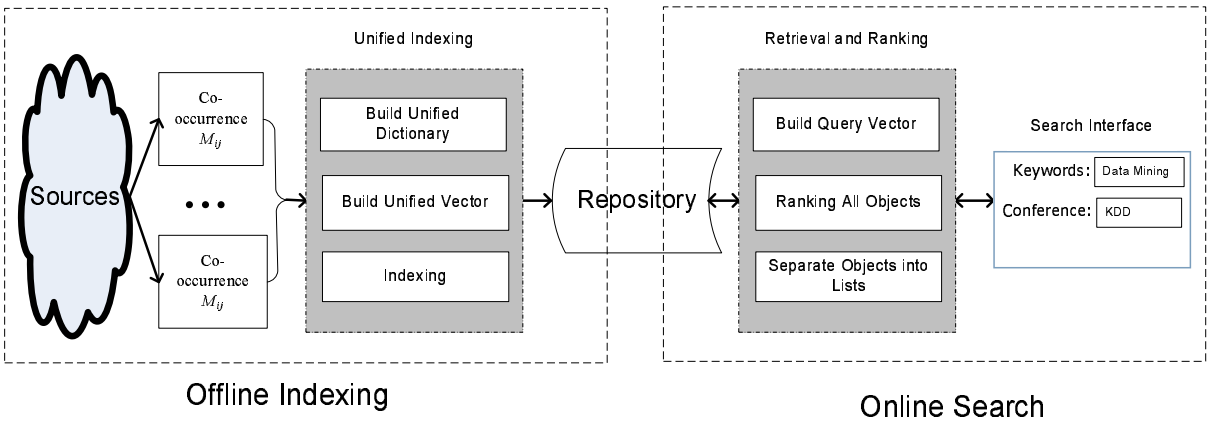

Figure 2: The system architecture of SHINE

Figure 3: The interface snapshot of SHINE for literature search.

Researchers	Conferences	Keywords
W. Bruce Croft	SIGIR	retrieval
James P. Callan	TREC	information
Chris Buckley	CIKM	document
Norbert Fuhr	ECDL	system
James Allan	ECIR	model
Gerard Salton	Hypertext	query
Clement T. Yu	DEXA	text
ChengXiang Zhai	VLDB	base
Mark Sanderson	EDBT	search
Wei-Ying Ma	WWW	trec

Table 2: Results for SHINE-query "Conference:-SIGIR"

Researchers	Conferences	Keywords
ChengXiang Zhai	SIGIR	retrieval
David A. Evans	TREC	model
Xuehua Shen	CIKM	information
Xiang Tong	KDD	trec
John D. Lafferty	ANLP	language
Tao Tao	ACL	clarit
Natasa Milic-Frayling	UAI	experiment
Hui Fang	ICME	text
Bin Tan	WebDB	feedback
Rong Jin	CSB	document

Table 3: Results for SHINE-query "Author:-ChengXiang Zhai"

6.1.2 Experiment Results

The interface for literature search is shown in Figure 3. This interface allows users to input SHINE-queries with different types of objects and the system can answer users' queries with multiple types of objects. These functionalities can satisfy several interesting information needs such as "What is the most famous researchers in SIGIR conference?" "What are the representative keywords to describe a researcher? " and etc.

We show several representative cases in Table 2 and Table 3. Due to the space limit, we did not show the returned papers in these results. In Table 2, our query is "SIGIR" of conference type. We can see the first researcher is "Bruce Croft" and the relevant words include "information", "retrieval", and "query". All these words accurately describe the research focus of "SIGIR" conference. The similar conferences to SIGIR are "TREC", "CIKM", and "ECIR", etc. All these are very meaningful. In Table 3, we show the results when the SHINE-query is "ChengXiang Zhai" of author type. We can see that all the returned words and conferences provide useful information to know the research interests of this researcher. This confirms the effectiveness of our solution to SHINE. Furthermore, SHINE can also facilitate users' information exploration. For example, a user may first issue "SIGIR" as a query along the conference dimension. After reading the results, she may want to know

Text-queries	Web pages	Keywords
xbox 360	www.microsoft.com/xbox	xbox
xbox	xbox360.teamxbox.com	360
xbox.com	www.teamxbox.com	game
www.xbox.com	www.xbox.com/en-us	xbox360
microsoft xbox 360	www.xbox.com	microsoft
xbox 360 news	xbox360.ign.com	box
xbox 360 locator	www.xbox.com/live	cheat
buy xbox 360	www.xbox.com/en-us/games	link
xbox 360 in stock	xbox360.1up.com	online
xbox 360 in stock now	forum.teamxbox.com	news

Table 4: SHINE results for "xbox 360" of text-query type

Text-queries	Web pages	Keywords
map quest	www.mapquest.com	map
mapqust	findme.mapquest.com	mapquest
wwwmapquest.com	maps.yahoo.com	direct
www.mapqwest.com	www.mapquest.ca/directions	quest
map quest.com	yp.mapquest.com	ca
www.mapquest.com/	www.mapquest.com/maps	mapqest
mapqest	maps.google.com	online
mapquest.com	www.mapquest.ca/maps	city
www.mapquest.com	www.randmcnally.com	mapqust
mapquest	www.multimap.com	travel

Table 5: SHINE results for "www.mapquest.com" of Web page type

more about researcher "ChengXiang Zhai" and issue this as a SHINE-query with author type. All these can be easily satisfied by SHINE and the results will change from Table 2 to Table 3.

6.2 SHINE Search on Click-Through Logs

Search engine logs contain useful information about the interactions between Web users and search engines, and have been studied intensively for various applications. For example, search engine companies want to provide better services for advertisers in order to help them bid high quality advertising keywords; Web masters hope to provide personalized contents to their users; and general users want to get assistance to formulate better queries. In this section, we apply SHINE to click-through log data to satisfy the above needs.

6.2.1 Data Set

Search engine logs generally have the following information: at what time, which user submit what query and visit what pages [24]. In our experiment, we do not consider the time information. For privacy concern, we do not include user information either. Thus we have 3 types of objects in this experiment: text-queries, Web pages, and words. The multiple-type graph in this application is a triangle with 3 vertices and 3 edges.

Our click-through log data is a sample from one month's log data of a commercial search engine. After preprocessing, we have 1,161,248 text-queries, 2,044,147 pages, and 252,102 words. In Figure 1, their exists a *containing* relation between pages and words. An intuitive way to build this relation is to use the contents of pages. In order to avoid crawling all the Web pages, we build this relation by pooling all the text-queries which refer to a certain page together as the pseudo-content of this page. Therefore, the log data is enough for us to build all the 3 co-occurrence relationships in this application. We again set all α_{ij}'s to 1.

6.2.2 Experiment Results

The interface for this application is similar to the one shown in Figure 3. Users can also exploit this search engine to answer various interesting questions. As illustrated in Table 4 and Table 5, given the input "xbox 360" of text-query type, the top returned text-queries include "xbox", "microsoft xbox 360", "xbox.com", and etc; the top returned Web pages include "www.microsoft.com/xbox" and "www.xbox.com", etc. Given the input "www.mapquest.com" of Web page type, the related text-queries, Web pages, and keywords are shown in Table 5. All these show the effectiveness of SHINE and can indeed facilitate the information needs of general users, Web masters, and advertisers as we discussed above.

Text-query type	Web page type	Word type
xbox 360	www.microsoft.com	hotel
ibm	www.ibm.com	toyota
digital camera	www.dell.com	weather
hotel	www.yellowpages.com	perl
free samples	www.yahoo.com	java
harry potter	www.xbox.com	depression
pizza hut	www.google.com	harry potter
depression	www.msn.com	music
free music	www.uiuc.edu	yellow page
toyota	www.mapquest.com	pizz hut

Table 6: 30 SHINE-queries composed from search logs: 10 with text-query type, 10 with Web page type, and 10 with word type.

In order to evaluate SHINE's search accuracy, we select 30 SHINE-queries from our log data: 10 of text-query type, 10 of Web page type, and 10 of keyword type. All these SHINE-queries are listed in Table 6. For each SHINE-query, we have 3 lists of objects in SHINE search results (e.g., Table 5). To evaluate these results, human subjects are asked to judge each returned object given a SHINE-query. We use precision at N documents (P@N) and let N range from 1 to 10 for evaluation. The results are shown in Figure 5. In this figure, the "Average" line corresponds to the evaluation on all the 90 groups of search results. "Text-query type input", "Web page type input", and "Word type input" correspond to the evaluation result when text-queries, Web pages, and words are used as input type respectively. From Figure 5, we can see that our solution to SHINE is quite promising: The average P@1 is 93.1% and P@10 is 75.8%. For SHINE-queries with different types, most of P@N's are larger than 70.0% and this shows that E-VSM is an effective model to handle different SHINE-queries.

6.3 SHINE Search on Recommendation Data

In this section, we show that traditional Collaborative Filtering (CF) can also be modeled in our SHINE framework. In CF, there are two types of objects: n users and m items. The task of CF is to recommend items to an active user based on a historical user by item rating matrix $R = [r_{ij}]$ with r_{ij} being the rating value of user i to item j. In the SHINE framework, we regard CF as to retrieve (recommend) items given a user as SHINE-query. We will show that our SHINE framework can model user-based CF [13], item-based CF [20], and also their combination. In this paper, we use SHINE-CF to denote CF in our SHINE framework.

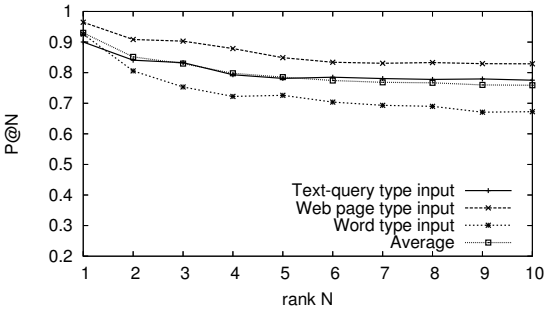

Figure 5: P@N of SHINE results on click-through logs with 30 SHINE-queries.

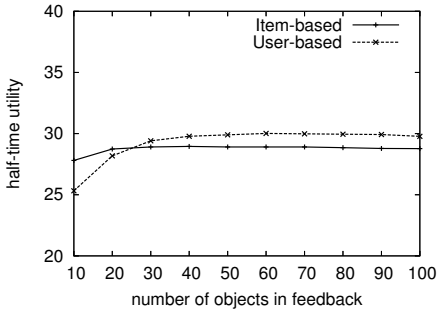

Figure 6: The impact of number of feedback objects in SHINE-CF

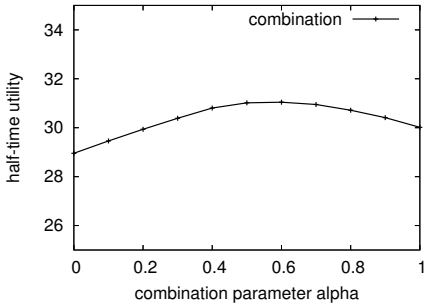

Figure 7: The impact of combination parameter α in SHINE-CF

6.3.1 Traditional CF

Given an active user, user-based CF first finds its similar users and then recommends items based on these similar users [13]. The most popular method to calculate the similarity between two users v and u is Pearson correlation:

$$w_{v,u} = \frac{\sum_{i=1}^{m}(r_{v,i} - \bar{r}_v)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{v,i} - \bar{r}_v)^2 \cdot \sum_{i=1}^{m}(r_{u,i} - \bar{r}_u)^2}} \quad (13)$$

where m is the number of items, $r_{v,i}$ ($r_{u,i}$) is the rating of user v (u) for item i, and $\bar{r}_v$ ($\bar{r}_u$) is the average rating of user v (u). $w_{v,u}$ is the similarity score between the two users. We use N_v to denote the set of selected nearest neighbors for v, then the prediction of v's rating for an unseen item i is calculated as

$$\hat{r}_{v,i} = \bar{r}_v + \frac{\sum_{u \in N_v}(r_{u,i} - \bar{r}_u) \cdot w_{v,u}}{\sum_{u \in N_v} w_{v,u}} \quad (14)$$

and the items with the largest prediction values are recommended to user v.

Item-based CF [20] is to first find similar items for each of the items that the active user rated. Then the recommendation score of a given item i is computed as

$$\hat{r}_{v,i} = \frac{\sum_{x:\text{ similar to } i \bigwedge v \text{ rated } x} w_{x,i} \cdot r_{v,x}}{\sum_{x:\text{ similar to } i \bigwedge v \text{ rated } x} w_{x,i}} \quad (15)$$

6.3.2 SHINE-CF

The Pearson correlation is indeed the cosine score between two vectors if we transform $r_{v,i}$ of R by $r_{v,i} - \bar{r}_v$, and thus obtain the matrix $\tilde{R}$. We compose the unified matrix in SHINE as

$$U = \begin{bmatrix} I & \tilde{R} \\ \tilde{R}^T & I \end{bmatrix} \quad (16)$$

Given a user as a SHINE-query, SHINE-CF is to retrieve both similar users and similar items with cosine similarity. It is easy to verify that in the unified space defined in Equation (16), the similarity between two users v and u is the same as $w_{v,u}$ in Equation (13). The retrieved items are those which have been rated by the current user. In the second step, we do the feedback to modify the SHINE-query using Equation (11) and then only retrieve items. If we only use retrieved users in feedback (denoted as user-based method in the following), SHINE-CF will be the same as user-based CF. If we only use retrieved items in feedback (denoted as item-based method in the following), SHINE-CF will be similar to item-based CF. Apparently if we use both users and items in feedback, we can combine these two methods together. In the following experiment, we will show that the combination can help improve the recommendation accuracy.

We use half-time utility metric to evaluate the recommendation accuracy [5]. For a user v, the expected utility of a ranked list of items is:

$$\mathcal{R}_v = \sum_j \frac{\max(r_{v,j} - \bar{r}_v, 0)}{2^{(j-1)/(\tau-1)}}$$

where τ is the half-time parameter ($\tau = 5$ in our experiments) and $r_{v,j}$ is v's rating for the item which is at the j-th position in the current rank list. The final score over all users in the test set is:

$$\mathcal{R} = 100 \frac{\sum_v \mathcal{R}_v}{\sum_v \mathcal{R}_v^{\max}}$$

where $\mathcal{R}_v^{\max}$ is the maximum possible utility obtained when all test items are ranked at top according to user v's rating.

We use the benchmark MovieLens[7] data set to compare different methods. Figure 6 shows the impact of number of objects used in feedback on both item-based method and user-based method in SHINE-CF. The best result of user-based method is achieved when the number of feedback objects is 60 and the best result of item-based method is achieved when the number is 40.

In Figure 7, we combine both item-based and user-based method by a parameter α and we vary α from 0.0 to 1.0 with step 0.1. When $\alpha = 0$, it is the item-based method and when $\alpha = 1$, it is the user-based method. Clearly, the

[7]http://www.grouplens.org/

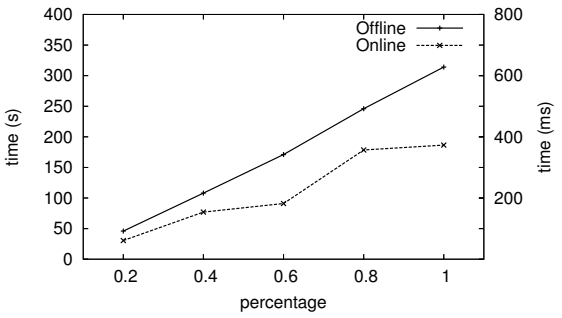

Figure 8: The offline indexing and online searching time with respect to the data size

combination can improve the recommendation utility. We get the best result when $\alpha = 0.6$. The combination method achieves relative improvement 7.2% over item-based method and 3.5% over user-based method.

6.4 Efficiency Evaluation

In this section, we study the efficiency of our SHINE solution. In particular, we study the average offline indexing and online searching time in our E-VSM model.

In this paper, we study the scalability of SHINE and report the experiment results on the search engine logs. We preprocess a large scale data set which contains 2,643,752 queries, 4,275,622 Web pages, and 6,888,081 words. In order to investigate the scalability of our method, we randomly select a certain percentage of the objects (from 20% to 100% with step=20%) from the whole data set to measure the average offline indexing and online searching time. The results are shown in Figure 8. We measure the offline indexing time by second and the online searching time by millisecond. The online time is the average time of 500 SHINE-queries randomly selected from our search logs. From Figure 8, we can see that the offline indexing time is linearly increasing with the data size. The online time is approximately sublinear. For example, when we increase the data size from 80% to 100% (25% relative larger), the average time increases from 357 to 373 ms (5% relative slower). These confirm that E-VSM is an efficient solution to the SHINE problem.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we define and study a novel search problem: search heterogeneous interrelated entities (SHINE). Compared with traditional search services, both the query capability and the search result richness are extended in our SHINE framework. We discuss and compare two methods, M-LSA and E-VSM, in this paper. Experiments with E-VSM on three data sets (a literature data set, a search engine log data set, and a recommendation data set) show the effectiveness and flexibility of our SHINE framework.

SHINE is a promising framework and there are several natural future work. First, we only focus on the query dependent relevance based ranking in this paper. In the future, we will study how to better combine it with link-based static ranking to further improve the search results. Second, we set matrix weights α_{ij}'s to 1 in our experiments. It would be interesting to study how to set α_{ij}'s automatically and a compelling method that we will study is to use machine learning techniques to learn these weight parameters.

8. REFERENCES

[1] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *KDD*, pages 14–23, 2006.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] H. Bast and D. Majumdar. Why spectral retrieval works. In *SIGIR*, pages 11–18, 2005.

[4] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *ICML*, 2005.

[5] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[7] C. Buckley, J. Allan, G. Salton, and A. Singhal. Automatic query expansion using smart: Trec 3. In *In Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 69–80, 1995.

[8] K. Chakrabarti, V. Ganti, J. Han, and D. Xin. Ranking objects based on relationships. In *SIGMOD Conference*, pages 371–382, 2006.

[9] S. Chakrabarti, K. Puniyani, and S. Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *WWW*, pages 717–726, 2006.

[10] T. Cheng, X. Yan, and K. C.-C. Chang. Entityrank: Searching entities directly and holistically. In *VLDB*, 2007.

[11] B. D. Davison. Toward a unification of text and link analysis. In *SIGIR*, pages 367–368, 2003.

[12] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology (JASIS)*, 41(6):391–407, 1990.

[13] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[14] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[15] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[16] B. Long, X. Wu, Z. M. Zhang, and P. S. Yu. Unsupervised learning on k-partite graphs. In *KDD*, pages 317–326, 2006.

[17] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: bringing order to web objects. In *WWW*, pages 567–574, 2005.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, November 1999.

[19] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval SystemExperiments in Automatic Document Processing*, pages 313–323, 1971.

[20] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[21] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[22] J. Wang, H.-J. Zeng, Z. Chen, H. Lu, L. Tao, and W.-Y. Ma. Recom: reinforcement clustering of multi-type interrelated data objects. In *SIGIR*, pages 274–281, 2003.

[23] X. Wang, J.-T. Sun, Z. Chen, and C. Zhai. Latent semantic analysis for multiple-type interrelated data objects. In *SIGIR*, pages 236–243, 2006.

[24] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, pages 87–94, 2007.

[25] J.-R. Wen, J.-Y. Nie, and H. Zhang. Clustering user queries of a search engine. In *WWW*, pages 162–168, 2001.

[26] W. Xi, B. Zhang, Z. Chen, Y. Lu, S. Yan, W.-Y. Ma, and E. A. Fox. Link fusion: a unified link analysis framework for multi-type interrelated data objects. In *WWW*, pages 319–327, 2004.