Web Object Indexing Using Domain Knowledge*

Muyuan Wang
Department of Automation
Tsinghua University
Beijing 100084, China
(86-10)51774518
wmy99@mails.tsinghua.edu.cn

Zhiwei Li, Lie Lu, Wei-Ying Ma
Microsoft Research Asia
Sigma Center, Haidian District
Beijing 100080, China
(86-10)62617711
{t-zli, llu, wyma}@microsoft.com

Naiyao Zhang
Department of Automation
Tsinghua University
Beijing 100084, China
(86-10)62787079
zlh@tsinghua.edu.cn

ABSTRACT

Web object is defined to represent any meaningful object embedded in web pages (e.g. images, music) or pointed to by hyperlinks (e.g. downloadable files). Users usually search for information of a certain 'object', rather than a web page containing the query terms. To facilitate web object searching and organizing, in this paper, we propose a novel approach to web object indexing, by discovering its inherent structure information with domain knowledge. In our approach, Layered LSI spaces are built for the hierarchically structured domain knowledge, in order to emphasize the specific semantics and term space in each layer of the domain knowledge. Then, the web object representation is constructed by hyperlink analysis, and further pruned to remove the noises. Finally, the structure attributes of the web object are extracted with the knowledge document that best matches the web object. Our approach also indicates a new way to use trust-worthy Deep Web knowledge to help organize dispersive information of Surface Web.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *Data Mining*; I.7.m [**Document and Text Processing**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Clustering, Selection Process.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Web object, indexing, domain knowledge, latent semantic indexing, link analysis, confidence propagation, information retrieval

1. INTRODUCTION

Contemporary web search engines are mainly 'page oriented', that is to say, their indexing granularity is web page. As a result, they are only able to provide search results in the form of ranked web pages with respect to user's query. However, in many cases, the users want to search for information of a certain 'object' rather than the web page. For example, users may use query "artist Beatles" to search for the biography about Beatles band, their albums and songs, instead of the web pages that contain the query terms only. To meet such information need, an 'object' oriented search engine is required. Consequently, it is expected that some integral technique could be developed to index the web objects.

In this paper, 'web objects' are defined to represent those meaningful objects embedded in web pages (e.g. images), or pointed to by hyperlinks (e.g. song streaming, downloadable files). Usually, the surrounding texts (including anchor text) can preliminarily describe a web objects. Complementary information of the web object may be also possible presented in the neighboring pages that have hyperlinks among them. Figure 1 illustrates two examples of web object, with or without descriptions in the surrounding texts. Figure 1(a) stands for a 'book' object, with a little valuable information in the surrounding text. The information about its author, publisher, and the introduction of the author are found in the pages hyperlinked with it. Figure 1(b) illustrates a 'song' object with the descriptions of its containing album and performer in the neighboring pages. In a broad sense, almost everything on the web can be regard as some kind of web object, including those virtual objects or concepts described in web pages (e.g. a music review or book review).

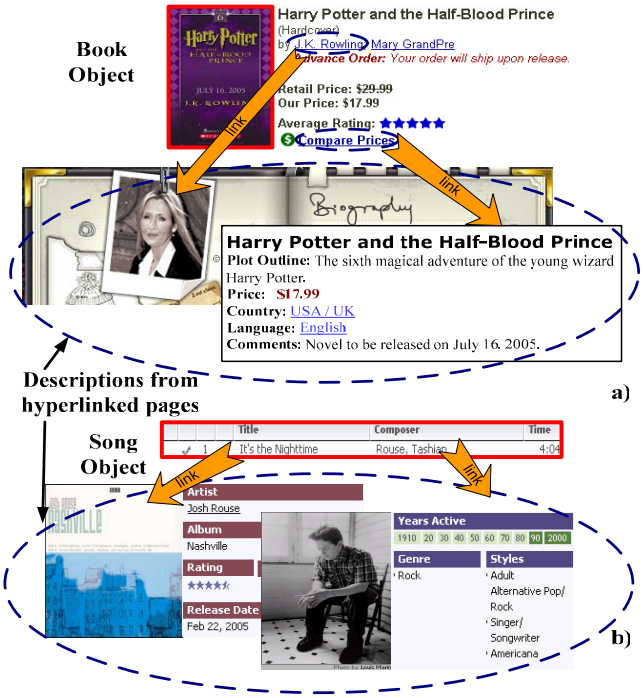

Figure 1. Some examples of 'web object'

Web objects usually lie in some implicit structure organizations. For example, song object can constitute a layer in the hierarchical structure of artist-album-song, as Figure 2 shows. Structured organization of web objects provides a good directory to facilitate user to browse web resources distributed dispersively.

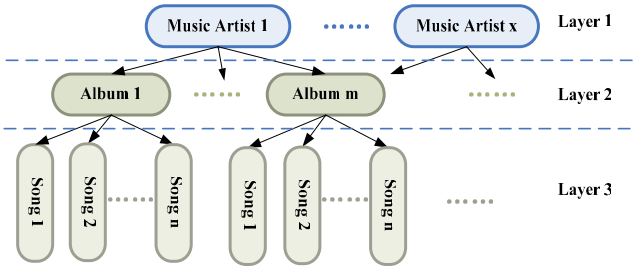

Figure 2 Hierarchical organization of a music database, which includes three layers: artist, album and song. In some Deep Web sites, each node is a web page; relationships between them are presented by hyperlinks.

To facilitate web object searching and organizing, in this paper, we propose a general framework to index web objects, and further, organize them based on their implicit structure.

To achieve such an objective, we should solve two major difficulties,

1. Lack of information. Usually, there is little descriptive textual information around web objects. The information may be noisy and insufficient to represent web objects. It is critical to enrich the description of the target web object and prune the noisy information.

2. Difficult to identify structure. Even if the description of a web object is sufficient and extracted, it still leaves a difficult problem to extract the structure information of the web object. Automatic discovery of implicit structure is very challenging. Although some hierarchical clustering methods are proposed to detect semantic structure, the obtained hierarchy is usually not meaningful to the target web object that is domain specific [15]. it will also meet some traditional obstacles like polysemy and synonymy due to various word usages in different web pages. Moreover, the traditional wrapper-based information extraction approaches are not suitable in these cases, since a wrapper is usually suitable to only one kind of web pages but could not adapt to all kinds of web pages, which are usually diverse in their design format.

To deal with these problems, we propose a novel non-clustering and non-wrapper approach to index web objects in various web pages, by integrating link analysis and domain knowledge. Hyperlink analysis is used to enrich the textual description of the web object; and domain knowledge is used to help noise removing and structure attributes construction of the domain-specific web objects.

Currently, many domain knowledge databases are available and well organized in Deep Web sites, such as All Music Guide (for music) [8] and Amazon (for books), where the domain knowledge is usually organized hierarchically, and each node is described by an individual web page. These databases provide meaningful and domain-specific structures for web objects, and thus can be greatly helpful in web object indexing. With hierarchical domain knowledge, web objects can be indexed using the structure information contained in the knowledge document that best matches the web object. From this point of view, our proposal provides a promising way to use trust-worthy Deep Web information to help information organization of Surface Web.

The rest of this paper is organized as follows: Section 2 presents the proposed framework. Section 3 describes our layered indexing scheme for hierarchical domain knowledge. Section 4 proposes our approach to web object representation through text and link analysis. Section 5 presents the matching process between web objects and the documents in knowledge database. Finally, the proposed approach is evaluated in Section 6, and an example application in music domain is presented in Section 7. Related work and conclusions are given in Section 8 and 9.

2. FRAMEWORK

Figure 3 illustrates our proposed framework for web object indexing. It is mainly composed of three steps: knowledge space building, web object representation, and web object indexing.

Step One: Knowledge Space Building. In the first step, the domain knowledge is indexed by Latent Semantic Indexing (LSI), for further use in web object representation and identification. A knowledge database is usually hierarchical or tree-like (as shown in Figure 2), where different layers have different semantics. Therefore, it is a better choice to represent knowledge layer by layer. Thus, in this step, knowledge is indexed in each layer respectively, and Layered LSI spaces are built for the domain knowledge.

Step Two: Web Object Representation. In this step, textual information of the web object is extracted and pruned. Firstly, a preliminary neighborhood graph is constructed around target web object with hyperlink analysis, in order to enrich the description of the web object. Then, the content of each web page is projected to the Layered LSI spaces of domain knowledge, to remove the irrelevant words to the web object. In order to obtain a better web object representation, neighborhood graph is further pruned to remove noisy pages. After pruning, all web pages in the pruned neighborhood graph are combined together to get a new description vector of the target web object.

Step Three: Web Object Indexing. The major task in this step is to identify the structure attributes of target web object based on its description vector and domain knowledge. To accomplish this task, the similarity between the description vector of web object and each knowledge document is firstly measured in the Layered LSI spaces; and then a process of structure-based confidence propagation is performed to select the knowledge document which best matches the web object. The corresponding structure attributes in the knowledge document are then used to index the web object.

Although the proposed framework emphasizes on utilizing the domain knowledge and the corresponding hierarchical structure, our approach is still feasible for non-hierarchical structure, which can be considered as a special case of hierarchical structures, with the layer number equal to one. In this simplified case, our approach becomes equivalent with the method proposed in [4].

[*] This work was performed when the first author was a visiting student at Microsoft Research Asia

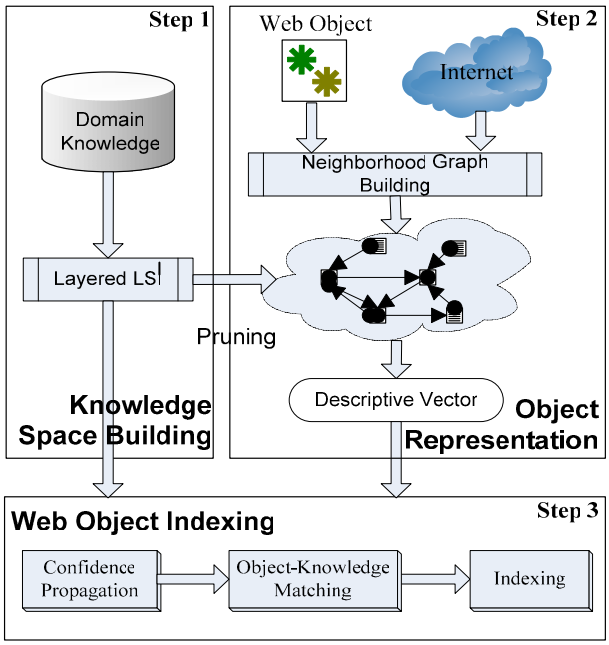

Figure 3 The framework of two-step indexing engine

It is noted that, in this paper, we mainly focus on web objects indexing, and assume web objects have been detected. In practice, web object detection is usually domain specific and thus can be solved with some simple heuristic rules.

3. KNOWLEDGE SPACE BUILDING

Both web object descriptions and knowledge database are textual, but usually are authored by different authors. This fact leads to their different term spaces. Consequently, directly using domain knowledge to identify web object is not feasible. In order to solve this problem, in our approach, Layered Latent Semantic Indexing is used to index documents in the knowledge database. LSI is capable to deal with polysemy and synonymy problems in some degree, as indicated in many textual retrieval applications [6].

3.1 Layered LSI of Knowledge Database

Usually, each layer of the hierarchically structured knowledge represents different concept, and has different term space. For instance, the three layers in music domain knowledge represent the concepts of artist, album and song, respectively; and the term 'Beatles' is obviously more probable to appear at the 'artist' layer, but not at the layer of 'album' or 'song'. Moreover, the scale of the term space of each layer is also different. Thus, if all layers are indexed together into one LSI space, the layer with smaller term space will be overwhelmed by the larger one. That means the term importance in the corresponding layer will be unfairly reduced. Therefore, in our approach, we index each layer independently and thus compose Layered LSI spaces.

In our approach, the textual content contained in each record or page of the knowledge database is considered as a *knowledge document*. These documents are tokenized to construct a term-document co-occurrence matrix. To improve the effectiveness of LSI, TFIDF is used as the weighting function in the co-occurrence matrix instead of term frequency only. In our approach, the simplest TFIDF formula is employed, as:

$$TFIDF(w) = f(w) \cdot \log \frac{N}{|D(w)|} \quad (1)$$

where w denotes a term, $f(w)$ represents the term frequency, N is the dimension of term space, and $D(w)$ stands for the set of documents that contain term w.

Supposing there are c layers in knowledge database, and the weighted term-document matrix in each layer is denoted as A_i, ($i = 1,\ldots,c$), each matrix can be decomposed by SVD as

$$A_i = U_i \Sigma_i V_i^T \quad (2)$$

Then, largest k singular values are selected to construct the latent semantic structure of A_i, denoted as A_i^k:

$$A_i^k = U_i^k \Sigma_i^k V_i^{kT} \quad (3)$$

Thus, each document (or query) q can be represented as $q^T U_i^k \Sigma_i^{k-1}$ in the new semantic space.

3.2 Discussion

Domain knowledge will be further used in the web object representation and web object indexing. In these modules, all web pages (describing the target web object) are mapped to the Layered LSI spaces, in order to remove irrelevant words that are out of domain. As mentioned above, each layer of domain knowledge has its unique semantic and term space. Thus, when we project each page into the LSI space of one layer, in some degree, only the words that are accord with the corresponding semantics are amplified; while other irrelevant words are suppressed or removed. Therefore, in the further processing, the similarity measure between two web pages or between web object and knowledge document, is computed in the Layered LSI spaces. It is more reasonable than direct comparison in the original space.

Meanwhile, such similarity comparison actually is measured in the LSI space of each layer, respectively. That is, the comparison is made from different semantic aspects. The basic idea behind this process is that we can use the information from all kinds of aspects to cross-verify the final decision. Our experiments also indicate the effectiveness of this method.

4. WEB OBJECT REPRESENTATION

The preliminary information of the web object can be found in the web page containing it or the corresponding web block [15]. However, the information is usually insufficient to describe the web object and may also contain many noisy words. In order to improve the web object representation, a neighborhood graph around the web object is constructed with text and hyperlink analysis. Neighborhood pages are considered in this case since they usually can help verify or complement information about web object. In this section, we firstly present the algorithm on neighborhood graph building and then address how to prune the noisy pages, which are irrelevant to the target web object.

4.1 Neighborhood Graph Building

To build a neighborhood graph, the web page or the web block containing the target web object is taken as the graph centre, and

all the web pages are 'connected' according to their hyperlinks. The utilized approach to neighborhood graph building is based on the technique proposed by Harmandas [4]. That is, it constructs an undirected graph, where each web page is taken as a node and each hyperlink as an edge, as illustrated in Figure 4. In the following sections, the web page (or web block) containing the target web object is referred as *container* for simplicity, while the pages linked with the container are called *neighborhood pages*,

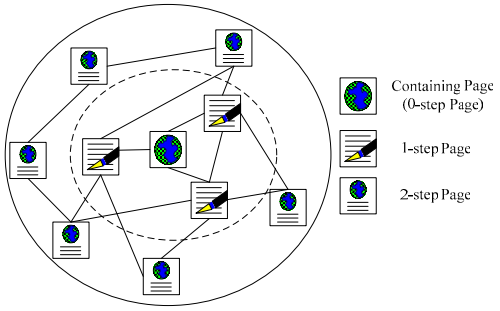

Figure 4 A 2-layer neighborhood graph with 1- and 2-step pages

In general, the less steps the page is from the container, the more closely the corresponding node is related to the target web object. Therefore, the number of the *Graph Layer* is an important parameter in web object representation. With larger neighborhood layer number, more information can be used to describe the web object, but more irrelevant (noisy) information is also introduced. In the experiments, we will discuss its selection.

4.2 Neighborhood Graph Pruning

The neighboring graph is assumed containing more information to describe the target web object, compared with the container itself. However, in the graph there are possible many noisy pages, which do not contribute to our task and will even lead to 'topic drift' [1], such as the '*contact us*' page and '*how to buy*' page. They are necessary to be removed for the further processing.

To remove the noisy pages, the similarity between each neighborhood page and the container should be calculated, assuming the pages similar to the container are more relevant to the web object.

In our approach, the cosine measure is used to measure similarity between each neighborhood page and container in each layer's LSI space. The similarity in the *j*th-layer actually represents the confidence that whether the corresponding page has mutual semantics with the container from the view of the *j*th-layer. Traditionally, the pages with small similarity can be considered as noisy pages [1]. In our experiments, a portion of pages with the smallest similarity in each layer is taken as noisy pages. The threshold is called *cutoff percent* in our paper and is determined experimentally. By this way, we get a set of noisy pages from each layer of domain knowledge, respectively. Since different pages have different semantics or functions, they may get different confidence in different layers. Therefore, only the intersection of these candidate sets is confirmed as noisy pages, and then pruned from neighborhood graph.

For some special cases where the knowledge database is not hierarchical, e.g., for that only consists of some plain text documents, our pruning scheme is still available with the layer number equal to 1.

Once removing noisy pages in neighborhood graph, we can integrate the textual information of all pages together to better describe the web object. In our approach, the weighted centroid (of neighborhood graph) is simply used as a description vector of web object, which is defined as,

$$C = \sum_{D_i \in G} w_i D_i \tag{4}$$

where G is the pruned neighborhood graph, D_i is a web page in G, and w_i is the weighting of web page D_i. The weighting is set according to the path steps from the corresponding web page to the container. A general rule is, the less steps the page is, the higher weighting it has. In our approach, the weighting is experimentally defined as equation (5) and then normalized to one,

$$w_i \propto \frac{1}{\log(d_i + 2)} \tag{5}$$

where d_i is the corresponding path steps, and the coefficient '2' is heuristically added in order to avoid zero in the denominator.

5. WEB OBJECT INDEXING

To this end, we get a description vector of the target web object in Layered LSI spaces. Then, the vector is used to extract more exact structure information of the target web object, by discovering the most appropriate knowledge document which best matches the web object. Thus, the problem is converted to textual information matching, that is, matching between the web object representation and each document in domain knowledge. At last, the obtained structure data in knowledge document will be used to index web objects.

5.1 Object-knowledge Matching

To achieve object-knowledge matching, the similarity between the description vector and each knowledge document in each layer is calculated using cosine measure. Usually, a web object belongs to only one certain layer of knowledge database. Therefore, our goal can be simply achieved by finding the *candidate document* that is most similar to the target web object, from the corresponding *target layer* in the knowledge database.

However, this method might be not accurate enough, since the similarity score only provides some confidence, and some noises would harm the matching accuracy. To deal with this issue, the hierarchical structure of domain knowledge is utilized, and each candidate document in the target layer is re-scored or re-ranked by confirming with its relatives (including ancestors and offspring). For example, as Figure 5 shows, when we re-rank the confidence of the node *T* (a candidate document), the confidence of its ancestors (node 1-2) and offspring (node 3-7) are all incorporated. This process is called *confidence propagation*. The confidences from the candidate document's relatives actually represent the similarities between the candidate document and the target web object in the corresponding semantic attributes (layers). The final score is thus cross-verified from various semantic aspects.

The *confidence propagation* from the relatives of a candidate document can be divided into two directions, propagating from its ancestors (root) nodes (top-down), and propagating from its offspring nodes (bottom-up), as Figure 5 illustrates. In top-down propagation, we directly sum the confidence score of up-layers in the path from root to candidate document. It is feasible since the

path from root to the candidate document is unique. However, from its offspring, the bottom-up propagation path is not unique. We could also sum scores of all paths from the candidate document to its offspring for confidence propagation. However, it is usually unfair since the candidate documents with lots of offspring would be over-weighted. To deal with this problem, the candidate document with a 'big' sub-tree is punished by dividing the number of its children, in our approach.

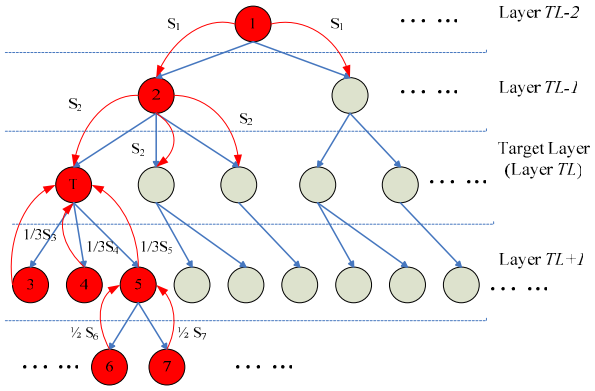

Figure 5 Web object matching and confidence propagation

The detailed formula of confidence propagation is shown in equation (6), which sequentially combines the information from upper and lower layers, as well as normalizes information coming from lower layers.

$$D(r_i) = s(r_i) + \sum_{k \in par(r_i)} D(k), \quad 1 \le i < TL$$

$$U(r_i) = s(r_i) + \sum_{k \in chi(r_i)} \frac{U(k)}{NC(r_i)}, \quad TL < i < L \qquad (6)$$

$$S(r_{TL}) = s(r_{TL}) + \sum_{k \in par(r_{TL})} D(k) + \sum_{l \in chi(r_{TL})} \frac{U(l)}{NC(r_{TL})}$$

where $s(.)$ represents the original confidence of a document, $S(.)$ means the updated confidence score of a document, $U(.)$ and $D(.)$ means the propagated confidence from bottom-up and top-down, respectively. r is a knowledge document and its subscript represents its layer number. L is the layer number of domain knowledge and TL is the target layer. $par(.)$ and $chi(.)$ stand for the set of parents and children of a document, respectively, and $NC(.)$ is the number of its children.

After confidence propagation, the candidate document with the highest similarity score $S(r_{TL})$ is chosen as the optimal document, which best describes the structure attributes of the target web object. The properties of this web object are further used in indexing and organization.

It is noted that, in the case that the knowledge database is not hierarchical, we need not do any confidence propagation.

5.2 Indexing Scheme

Once web object is matched with a knowledge document, the trust-worthy domain knowledge can be used to further index and organize web objects. It can be divided into two cases:

1. If the domain knowledge is a fully-structured database, the record attributes of the knowledge document can be used to represent the structure information of the web object and then index them. For example, if a song object is matched with a knowledge document, in which the corresponding metadata, such as artist, album, genre and release year, are available, these structure data can be further used to index the song object. Furthermore, many applications can be developed based on these structure indices.

2. If the domain knowledge is not a structure database, that is, the knowledge document is not a record but plain text, we can use the text to describe web object better. Then, traditional indexing scheme in text search/retrieval can be used to index the web objects.

6. EXPERIMENTS

In this section, we evaluate the performance of the proposed approach in the music domain, including,

1. We evaluate the overall performance of our approach, respect to various settings of neighborhood graph, such as the *graph layer* and *cutoff percent* mentioned in section 4. We also look into the performance on different web objects.

2. We compare our approach with the algorithm proposed in [4] (note that the algorithm in [4] is just a specific version of our approach, with the *cutoff percent* equals 0).

3. We evaluate the effectiveness of each module of our proposed framework, including domain knowledge indexing, neighborhood graph in web object representation, and confidence propagation in web object matching.

In order to measure the precision of our approach, the process of web object indexing is viewed as an IR problem. That is, the target web object is taken as a query, the knowledge documents are ranked according to their confidences, and then the top N are returned. Thus, the precision at top N results is used to evaluate the performance, as

$$P@N = \sum_{q_i \in Q} \left| q_i \cap R_i^N \right| \Big/ |Q| \qquad (7)$$

where Q is the query set and q_i is a query, R_i^N is the top N results corresponding to query q_i, and $|\cdot|$ is to get the count. In our experiments, $P@1$, $P@5$, and $P@10$ are used for performance evaluation.

All our algorithms are implemented in C++, and experiments are run on two workstations with two 3.0G Hz Intel CPUs each and 3.0G memory. For SVD, we use SVDPACK which can deal with large-scale sparse matrix [9][13].

6.1 Data Preparation

Music is one kind of most valuable objects on the web, and many mature music databases exist as Deep Web sites, which can be used in our experiments conveniently. Therefore, our approach is performed and evaluated in the music domain. In this section, we present the data preparation for domain knowledge and testing music objects, which are further divided into song objects, album objects and artist objects.

6.1.1 Domain Knowledge Preparation

In our approach, the domain knowledge about music is collected from All Music Guide [8], which is a Deep Web site. The artist pages, album pages and song pages are downloaded, and then organized in a three-layered, tree-like structure, in which each layer represents artist, album, and song, respectively, as

shown in Figure 2. The details of our collected domain knowledge are shown in Table 1. Overall, there are 32513 pages, including 26279 song pages, 4635 album pages and 1599 artist pages.

Type	#Documents	#Overall
Artist	1599	32513
Album	4635	
Song	26279	

Table 1 The details of our domain knowledge tree

A specific HTML parser is developed to re-build up the knowledge hierarchy from the crawled pages. The parser is used first to explore the implicit semantic relationships between pages; and second, to extract specific keywords, such as artist name, album title, and genre, for further web objects indexing and other applications.

After the knowledge is ready, LSI is used to index them. In text retrieval domain, many researchers reported that keeping 100 - 300 dimensions is a good trade-off between performance and computation complexity of LSI. In our experiments, the dimension is set as 200 accordingly. Based on our algorithms, three layers of the knowledge tree are indexed respectively. For comparison, we also used the non-layered LSI, by combining all documents of three layers into a whole corpus.

6.1.2 Music Objects Collection

As mentioned above, we mainly focus on web objects indexing, and assume web objects have been detected. Since there are few works address this task, in our experiments, web objects are detected with some simple heuristic rules. For example, a song object usually associates with some file-type indicators, such as file extension 'rm' or 'wma', and keywords 'listen'.

In order to collect the testing web objects, we first collect the top queries from search log of All Music Guide in one week, which includes hundred of songs, albums and artists, respectively. The queries not contained in our knowledge tree are removed in order that each potential music object has corresponding structure information in the domain knowledge. Then, we throw the valid queries into Google to obtain some related web pages, and crawl pages by using them as seeds. In the crawled pages, only those, which contain hyperlinks pointing to music files (with special extensions, such as rm, mp3, wma), are labeled as music object pages. Table 2 shows the details of our testing music objects.

A 2-layer neighborhood graph is crawled for each music object. We do not consider 3-step pages (in the third layer), since the web pages increase explosively but a majority of the pages are irrelevant to the target music object. We also limit the total page number under 500 in neighborhood graph building, in order to limit the computations.

Object Type	#Object	#Average pages
Artist	30	457
Album	50	460
Song	200	373

Table 2. The details of music objects in testing set

Finally, the ground truth is manually annotated for evaluations.

6.2 Overall Performance

In the first experiment, we evaluate the overall performance of the proposed approach on testing song objects. In summary, our approach is configured as following modules, domain knowledge representation using Layered LSI spaces, neighborhood graph building and pruning, and hierarchical web object matching. However, the performance is also related to two parameters, *graph layer* and *cutoff percent* in the construction of the neighborhood graph.

Figure 6 illustrates the system performance (P@5) on song objects indexing, corresponding to various graph layer and cutoff percents. In our experiments, the graph layer is chosen as 1 or 2, due to the explosive number of irrelevant 3-step pages; and the cutoff percent ranges from 25% to 100%, with an interval of 5%.

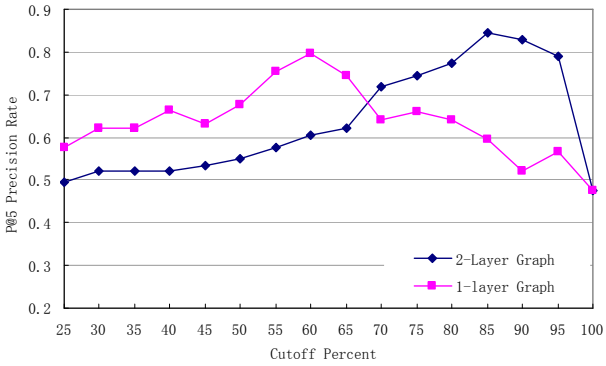

Figure 6 Overall performance respect to various graph layers and cutoff percents

From Figure 6, it can be seen that more than 80% web objects can be correctly found in the returned top 5 knowledge documents, with 60% cutoff in 1-layer graph and 85% in 2-layer graph. It can be also seen that, with the increase of cutoff percent, the performance is firstly increases and then decreases. It is reasonable, since when the cutoff percent is small, noisy information mainly affects the performance; while the cutoff percent is large, relevant information may be also filtered so that the precision decreases.

Moreover, the best performance using 2-layer graph is 4% better than that of 1-layer graph. However, a majority (85%) of pages in the 2-layer neighborhood graph are considered as noise and removed. Although it means some waste in page crawling and parsing, in the following experiments, we still uses 2-layer graph (including container, 1-step and 2-step pages) with optimal cutoff 85% as our default setting, in order to get higher performance.

In our experiments, we also find the performance of our approach is heavily depended on descriptive information of the song object. In order to look into the performance on different testing song objects, we manually classify them into three sets, including '*self-descriptive*', '*multi-page descriptive*' and '*non-descriptive*' object. The 'self-descriptive' object contains sufficient information in its container page for human to tell its structure property; and the 'multi-page descriptive' object have to be identified from the container combining with the neighborhood pages; while the 'non-descriptive' object does not contain sufficient discriminative information in its neighborhood graph.

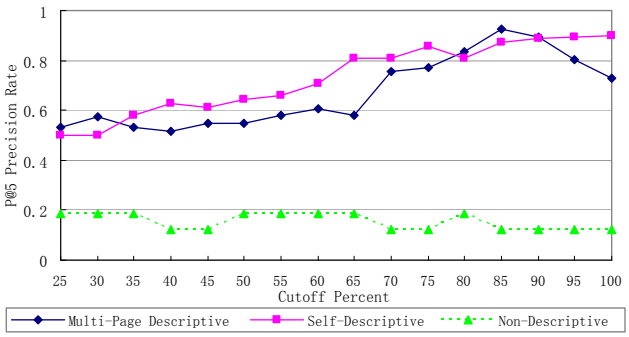

Figure 7 Precision curve of different set of song objects

Figure 7 illustrates the performances corresponding to different sets of song objects. From the figure, we find that the performance on self-descriptive set first increases and then almost keep the same when cutoff percent grows, while the performance on multi-page descriptive set has a increasing phrase and decreasing phrase, with optimal cutoff equal to 85%. It actually reflects the tradeoff between the noisy information and complementary information contained in the neighborhood pages. However, the performance of non-descriptive set is always poor. It is intuitive since no sufficient information is provided in the neighboring pages.

6.3 Look Into Each Module

The above section evaluates the overall performance of our proposed approach. In this section, we evaluate the effectiveness of each module which is corresponding to each step in section 2, including Layered LSI spaces for domain knowledge, neighborhood graph, and confidence propagation.

6.3.1 Domain Knowledge with Layered LSI Indexing

In our approach, domain knowledge is used, and Layered LSI spaces are built to represent domain knowledge, since different layer of domain knowledge usually has different meaning and term space. In order to evaluate the effectiveness of domain knowledge and Layered LSI, an experiment is designed to compare the methods with different usages of knowledge, in the module of neighborhood graph pruning (keep other modules the same). The compared methods include 1) Layered LSI on domain knowledge, 2) one-layer LSI on domain knowledge, and 3) without domain knowledge (that is, comparing pages in neighborhood graph directly [1]).

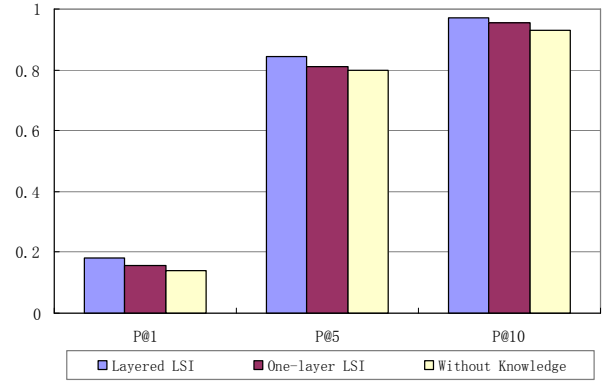

Figure 8 Precision comparison on different indexing approaches

The performance comparison of these three approaches is shown in figure 8. It is obvious that the performance of Layered LSI indexing is best. It is about 5% better than that without LSI, and about 3% better than one-layer indexing. It not only indicates knowledge is helpful in web object representation, but also indicates that layered indexing can obtain better results.

6.3.2 Neighborhood Graph for Object Description

In section 4, the neighborhood graph is built to enrich the description of the target web object, since the container page usually does not provide sufficient discriminative information. It is especially suitable for those *multi-page descriptive* objects. Figure 9 illustrates the performance of multi-page descriptive object indexing, comparing among 1) 1-layer neighborhood graph (with optimal cutoff 60%), 2) 2-layer neighborhood graph (with optimal cutoff 85%), and 3) without neighborhood graph (only the container page).

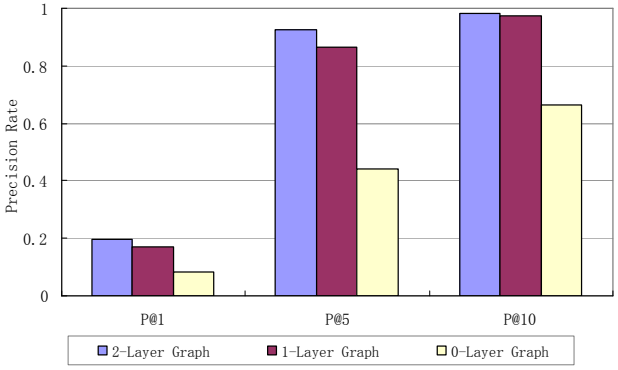

Figure 9 Performance comparisons with/without neighborhood graph for multi-page descriptive song objects

It can be seen that, with considering neighborhood pages, the performance is dramatically improved. For example, the P@1 and P@5 are almost doubled after using 1-layer graph or 2-layer graph.

6.3.3 Confidence Propagation for Object Matching

In Section 5, confidence propagation is performed to re-score the similarity between a candidate document and the target web object. After propagation, the confidence of candidate document is verified by confirming with its relatives, and consequently, better results are obtained. Figure 10 illustrates performance comparison between with confidence propagation and without confidence propagation, in the module of web object matching.

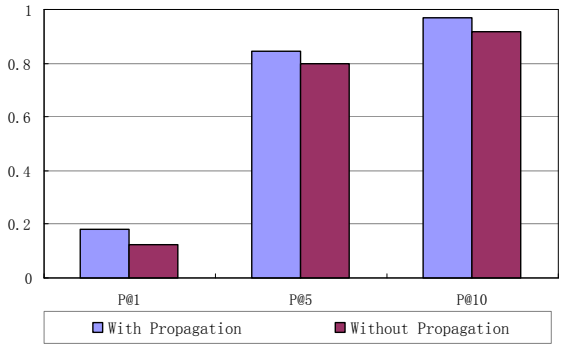

Figure 10 Performance comparisons with/without confidence propagation

It can be seen that, with confidence propagation, the precision (P@1, P@5 and P@10) is improved about 5%-7%. It indicates confidence propagation is also very helpful in web object indexing.

6.4 Performance on Different Music Object

To explore the performance on music objects in the same domain but located in different layers of the knowledge database, we also evaluate the performance on album objects and artist objects. The detailed comparisons are illustrated in Figure 11.

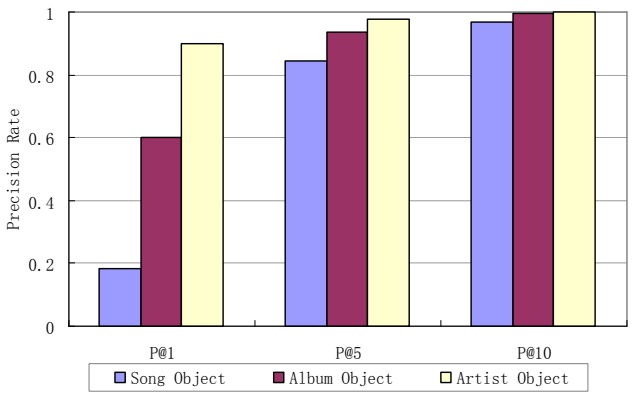

Figure 11 Performance on music objects in different layers

It is noted that, the performance on artist and album object is much better than song object, especially on P@1. There are two reasons. Firstly, due to the distinct page properties in music domain, the terms used to describe an artist or album is usually much more than a song. In other words, there is much more information that can be used to distinguish artists and albums. Secondly, information in song page is usually not discriminative. By combining with neighborhood pages, the description of a song object has a large overlap with those songs from the same album. As a result, it is usually difficult to discriminate the songs from a same album, while it is relatively easy to identify an album or artist object. Therefore, the P@1 of song object is much lower than that of albums and artists; however, for P@10, the values are very similar since an album usually has about 10 songs only.

7. APPLICATION: MUSIC SEARCH

The proposed technology on web object indexing can be used in various domains. To illustrate further use of indexed web objects, an example prototype on music search is developed in this section. The prototype is designed to organize numerous music resources dispersed on the Surface Web. With the obtained structured indices using our approach, the search results are organized according to their implicit structure. Moreover, all the related information, which is discovered from the knowledge documents, could also be associated with the search results and presented to users.

Figure 12 illustrates a snapshot of the prototype system. With the query 'yesterday', the returned results are organized according to their artist, album and year. As shown in the left panel of Figure 12, the different versions of 'yesterday' and similar songs were clustered to different artists, such as the Beatles, Boys II man and Cilla Black; and then they are further classified into different albums, such as 'Sessions' and 'Yesterday…and Today'. As our experiment shows, although it is difficult to identify a song, the accuracy of album and artist identification is relatively high. Therefore, organize the searching results into artist and album is feasible based on our proposed approach.

Moreover, the related information of current category, which are obtained from the matched knowledge documents, is shown in the right panel of Figure 12, including 'song details', 'album information' and 'track list'. The indexed web objects, which are obtained from the Surface Web and associated with the current category, are also listed in the 'Web Objects' block.

The prototype provides more related and organized information than contemporary web music search engines. The obtained structure is more meaningful in the domain than those obtained by traditional semantic clustering approaches [14].

8. RELATED WORK

Previous works on object-oriented indexing can be traced back to the framework of image retrieval proposed by Harmandas [4], in which a model for Web images retrieval is presented. The model is based on combining the text content and hypertext in neighborhood graph. However, their definition on objects is limited to web images, and they did not consider finding the underlying structures of web objects, either. There are also some efforts to obtain music structure information from Deep Web sites. For example, Windows Media Player 10 has the function to extract album information of a song from MSN Music, based on other metadata embedded in the ID3 tag.

Using a connective cluster of web pages to improve web search is also addressed in some work. Kleinberg's HITS [2] is an early important effort using link analysis to rank documents, while it only depends on in/out-degrees of links. Bharat et al. [1] identified a problem of HITS: *topic drift*, and proposed to resolve it by content analysis in local graph. This modification achieved a remarkable better precision in topic distillation application. By manipulating the weights of pages, Chang et al. [3] created customized authority by adding more weights to the documents that users are interested in. Recently, researchers proposed to use connective analysis to improve term weighting scheme in hyperlink environment [5]. However, none of them considered using domain knowledge to improve the combination of link analysis and content analysis.

Similar to our work (matching web object representation in knowledge database), "DB+IR" [10][11][12] was proposed in the database domain to use unstructured query to search structured database. Different from traditional SQL (Structure Query Language) technologies, some works in "DB+IR" regard database records as plain text, and use IR technologies to index and search database. Some researchers also proposed to improve search precision by using the structure of database, such as XSearch[12], which utilized the structure of XML DOM tree, and proposed to use ILF(Inverse Leaf Frequency) to punish common terms in leaf nodes, which was replacement of traditional IDF.

9. CONCLUSION

In this paper, we propose a novel approach to indexing web objects by using corresponding domain knowledge. Although in this paper our method is utilized and evaluated in the music domain only, we believe that our approach is feasible for general web objects indexing. Here we conclude our approach:

1. Domain knowledge from traditional databases or Deep Web sites could be used to index web objects to help build a domain-specific structure. This method overcomes the disadvantages of traditional approaches on hierarchical clustering and automatic ontology building.

2. Indexing hierarchical knowledge database layer by layer is a good strategy, which emphasizes each layer's concepts and suppresses noises effectively. Moreover, this method enables us to compare objects from various aspects (various attributes of object). It is an improvement of the traditional approaches using knowledge.

3. Confidence propagation can effectively utilize the relationships among nodes of knowledge tree. Each node's confidence can be amplified or suppressed with the confidences of its relatives. It actually also provides an enhancement for traditional IR methods.

In the future works, we will apply our approach into more domains to build efficient systems. We will also further develop a better approach to web object detection, and use technologies on web information extraction to obtain better description of the web objects.

10. REFERENCES

[1] K. Bharat, and M. R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In *Proceedings of the SIGIR conference on Information Retrieval*, 1998, pages 104-111

[2] J. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, Volume 46, Issue 5, pages: 604 – 632, 1999

[3] H. Chang, D. Cohn, and A. McCallum. Creating customized authority lists. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[4] V. Harmandas, M. Sanderson, and M. D. Dunlop. Image retrieval by hypertext links. In *Proceedings of the SIGIR conference on Information Retrieval (SIGIR '97)*, 1997

[5] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of TF-IDF Schemes for Web Pages using their Hyperlinked Neighboring Pages. In *Proceedings of the fourteenth conference on Hypertext and Hypermedia*, 2003

[6] S. Deerwester, S. Dumais, et. al. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41-6, 1990, pages 391-407

[7] Wordnet 2.0: A lexical database for the English language. *http://wordnet.princeton.edu/wn2.0*, 2003

[8] All Music Guide. *http://www.allmusic.com*

[9] B. Michael, D. Theresa, et. al. SVDPACKC (Version 1.0) User's Guide. *Technical Report: UT-CS-93-194*, University of Tennessee. 1993

[10] B. Y. Ricardo, and M. P. Consens. The continued saga of DB-IR integration. *The 30th International Conference on Very Large Databases (VLDB)* Tutorial, 2004

[11] V. Hristidis, and Y. Papakonstantinou. DISCOVER: Keyword Search in Relational Databases, In *Proceedings of the 28th International Conference on Very Large Databases (VLDB)*, 2002

[12] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In *Proceedings of the 29th International Conference on Very Large Databases (VLDB)*, 2003

[13] SVDPACK. *http://www.netlib.org/svdpack*

[14] B. Liu, C. W. Chin, and H. T. Ng. Mining Topic-Specific Concepts and Definitions on the Web. In *Proceedings of the twelfth international World Wide Web conference (WWW2003)*, May 2003

[15] D. Cai, S. Yu, J. Wen, and W.-Y. Ma. VIPS: a Vision-based Page Segmentation Algorithm, MSR-TR-2003-79, 2003

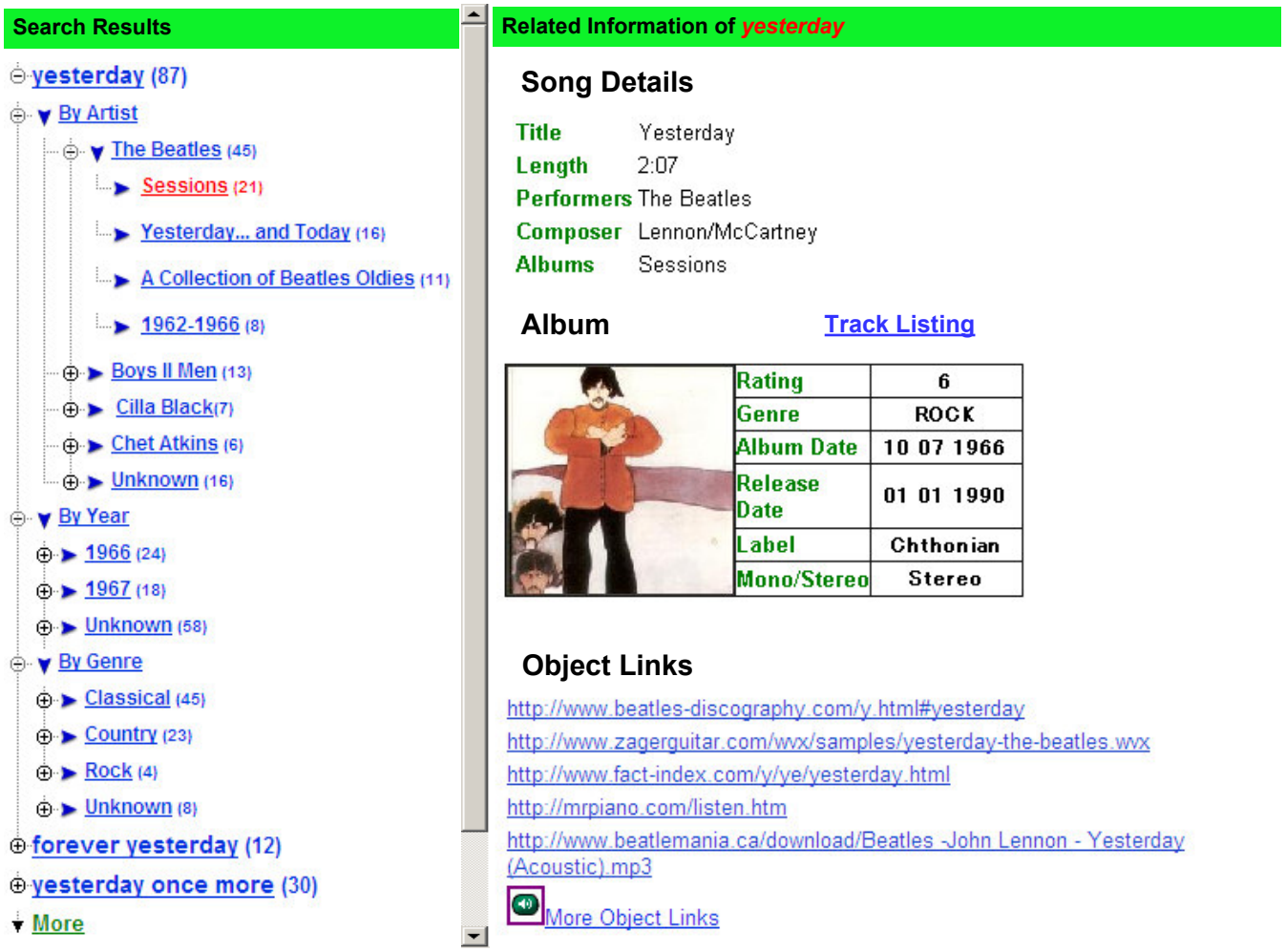

Figure 12 A snapshot of a prototype 'music search' system, utilizing the indexing created by our proposed approach