Object-Level Ranking: Bringing Order to Web Objects[*]

Zaiqing Nie
Web Search & Mining Group
Microsoft Research Asia
Beijing, P. R. China
t-znie@microsoft.com

Yuanzhi Zhang[†]
CS Dept.
Peking University
Beijing, P. R. China
zhangyzh@db.pku.edu.cn

Ji-Rong Wen Wei-Ying Ma
Web Search & Mining Group
Microsoft Research Asia
Beijing, P. R. China
{jrwen,wyma}@microsoft.com

ABSTRACT

In contrast with the current Web search methods that essentially do document-level ranking and retrieval, we are exploring a new paradigm to enable Web search at the object level. We collect Web information for objects relevant for a specific application domain and rank these objects in terms of their relevance and popularity to answer user queries. Traditional PageRank model is no longer valid for object popularity calculation because of the existence of heterogeneous relationships between objects. This paper introduces *PopRank*, a domain-independent object-level link analysis model to rank the objects within a specific domain. Specifically we assign a popularity propagation factor to each type of object relationship, study how different popularity propagation factors for these heterogeneous relationships could affect the popularity ranking, and propose efficient approaches to automatically decide these factors. Our experiments are done in the context of Libra, an object-level paper search engine indexing 1 million CS papers.

Keywords

Web Objects, Object-level Web Search, Link Analysis

1. INTRODUCTION

Existing Web search engines generally treat a whole Web page as the unit for retrieval and consuming. However, there are various kinds of objects embedded in the static Web pages or Web databases. Typical objects are products, people, papers, organizations, etc. We can imagine that if these objects can be extracted and integrated from the Web, powerful object-level search engines can be built to meet users' information needs more precisely, especially for some specific domains. Such a perspective has lead to significant interests in research communities, and related technologies such as wrapper deduction [11, 3], Web database schema matching [16, 8], and object identification on the Web [15] have been developed in recent years. These techniques made it possible for us to extract and integrate all the related Web information about the same object together as an information unit. We call these Web information units *Web objects*. Currently, few work has been done in retrieving and ranking relevant Web objects to answer user queries.

Like Web pages that are connected among each other to form a so-called Web graph, various kinds of objects also form an object graph based on the correlations among them. In the Web graph, different pages have different popularity according to their in-links. Technologies such as PageRank [13] and HITS [10] have been successfully applied to distinguish the popularity of different Web pages through analyzing the link structure in the Web graph. It is obvious that, in the object graph, objects are also not equally popular. Take the research domain as an example. Only several top conferences within a research field can attract high quality papers, and their papers are more likely to be read. In order to help users quickly locate their interested objects, we should calculate the popularity of collected objects. Because it is clear that the more popular the objects are, the more likely they will be interested by a user. So a natural question is: could the popularity of Web objects be effectively computed by also applying link analysis techniques? This paper targets to answer this question. Our answer to the question is yes, but quite different technologies are required because of the unique characteristics of object graph.

For link analysis, the most unique characteristics of the object graph is the heterogeneity of links, i.e., objects are related to each other through different types of relationships. For example, a paper object may be cited by a set of other paper objects, written by a set of author objects, and published in a conference/journal object (see Figure 1). So there are three kinds of different links in the graph: cited-by, authored-by and published-by and they have quite different semantics. The traditional link analysis methods including PageRank and HITS assume that all the links are with the same "endorsement" semantics and equally important, directly applying these methods would result in unreasonable popularity ranking. For example, the popularity of a paper should not be affected too much by the number of authors, and the number of citations does have a large impact on it. In this paper, we propose *PopRank*, a method to measure the popularity of Web objects in an object graph.

PopRank extends the PageRank model by adding a *popularity propagation factor (PPF)* to each link pointing to an object, and uses different propagation factors for links of different types of relationships. For example, for the links pointing to a paper object, we need three propagation factors (γ_3, γ_2, and γ_1 in Figure 1) for the three different types of relationships: cited-by, authored-by, and published-by,

[*]Technical Report. Microsoft Research. MSR-TR-2004-129
[†]This work is done when the author is visiting Microsoft Research Asia.

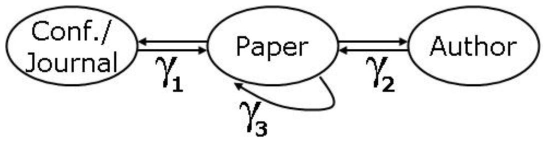


Figure 1: Paper Object Relationship Graph

respectively. However manually assigning these factors to make the popularity ranking reasonable is extremely challenging. With a huge link graph, it is very hard for us to tell which types of links are more important and even harder to quantify their exact importance. Fortunately it is always easier for us to collect some partial ranking of the objects from domain experts. For example, as a researcher, we know the order of the top conferences or journals within our field, and we may also know which papers are more popular. We propose a learning based approach to automatically learn the popularity propagation factors for different types of links using the partial ranking of the objects given by domain experts. The simulated annealing algorithm is used to explore the search space of all possible combinations of propagation factors and to iteratively reduce the difference between the partial ranking from the domain experts and that from our learned model.

One major challenging problem facing our learning approach is that it is prohibitively expensive to try hundreds of combinations of feasible factors which is normally needed for us to get a reasonable assignment of the propagation factors. Since it may take hours to compute the *PopRank* of the objects to test the optimality of a PPF factor assignment. In order to make the learning time manageable, we propose to use a subgraph of the entire object link graph in the learning process. Because as soon as we have most of the related objects and their links surrounding the training objects [1], we should be able to calculate an close approximation of the *PopRank* of these training objects. Since we are not interested in getting the exact rank scores but the relative rank of the these training objects, very little reduction of accuracy will not affect the optimality of the assignment too much. However in cases where the object link graph is prohibitively large, one might have to trade optimality for efficiency.

The *PopRank* link analysis model described and motivated in the foregoing has been fully implemented, and has been evaluated in the context of a paper search engine called *Libra*[2]. In this paper, we describe the details of the *PopRank* model, and its use in *Libra*. Our model could be also applied to many other vertical search domains including online product search, image search, music search, and people search with little modification. Our experimental results on *Libra* show that *PopRank* can achieve significantly better ranking results than naively applying PageRank on the object graph.

The rest of the paper is organized as follows. In the next section, we give a mathematical description of the *PopRank* link analysis model and provides intuitive justification. Section 3 describes the details of our link importance learning approach. Section 3.2 describes how we select a subgraph of the entire object link graph. In Section 4, we briefly introduce our Libra paper search. Section 5 describes the experiments we have done with *PopRank* and *Libra* to evaluate the effectiveness of our approach. Section 6 discusses related work and possible extensions, and Section 7 presents our conclusions.

[1]The objects ranked by domain experts

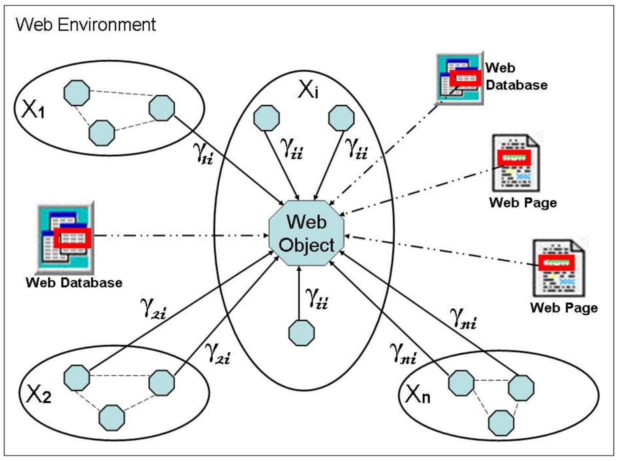


Figure 2: BackLinks of an Object of Type X_i from other Objects and the Web

2. THE POPRANK MODEL

Suppose we have n different types of Web objects X_1,X_2, ..., X_n in an application domain. For example, in the scientific literature domain, we have 4 types of objects: papers, authors, conferences, and journals. These objects are related to each other in different ways: a paper cites another paper, a paper is written by an author, a paper is published by a conference, and so on. In this section we give the details of our *PopRank* model for computing the popularity of the objects with the same type.

2.1 Web and Object Relationship Graph

In Figure 2, we show the back links of an object of type X_i. In the Figure, the solid arrows are used to show the back links from both the objects of the same type and those of other types, and the dotted arrows are used to show that the object is contained in a Web page or a Web database.

Since Web objects are contained in Web pages or Web databases, so the popularity of these Web pages and databases could also affect the popularity of their contained objects. If a Web pages is popular, its object information is more likely to be read. We use *Web popularity* to denote the probability that a ”random surfer” on the Web reads the information about an object by keeping clicking on the links between Web pages (including Web pages dynamically generated by Web databases). Since the information about an object is normally represented as a block of a Web page [12, 14]. The Web popularity can be computed by considering the PageRank [5, 13] scores of Web pages containing the object and the importance of the Web page blocks [14, 6]. We assume Web databases will uniformly propagate its popularity (i.e. PageRank) to their objects, so the Web popularity from these sources can be easily computed.

The relationship graph of the objects in a domain is another important resource that can be used to calculate a popularity ranking for an object. For example, assuming a reader wants to get into a new research field and to read the related papers. To get started, he may first use Google or CiteSeer to find several seed objects which could be some related papers, authors, or conferences/journals. After that he most likely just follows the object relationship links to locate more papers. He may want to read the papers cited by the papers he has already read, or read papers of his favorite authors, conferences, or journals. Clearly, a paper cited by a large number of popular papers could be popular, and a recent paper published in a prestigious conference with few citations could also be popular.

2.2 PopRank

We introduce a "random object finder" model to explain the reader's behavior. The "random object finder" simply keeps clicking on successive Web page links, Web page to object links, and object relationship links at random. Basically he starts his random walk on the Web, and he will start following the object relationship links once he finds the first object on the Web, never hitting "back" but eventually gets bored and will restart his random walk on the Web again to find another seed object.

We use a vector R_{EX} to denote the probability that the "random object finder" finds the object x only through the link graph of the Web, and another vector R_X to denote the probability that he finds the object by random walk both on the graph of the Web and on the object relationship graph.

To compute the popularity score of an object, the *PopRank* model takes into account both the Web popularity of the object and its relations with other object. We use the following formula to compute the *PopRank* scores R_X of the objects of type X:

$$R_X = \varepsilon R_{EX} + (1-\varepsilon)\sum_{\forall Y} \gamma_{YX} M^T_{YX} R_Y$$

Where

- $X = \{x_1, x_2, ..., x_n\}$, $Y = \{y_1, y_2, ..., y_n\}$: objects of type X and type Y;
- R_X, R_Y: vector of popularity rankings of objects of type X and type Y;
- M_{YX}: adjacent matrices,

  $m_{yx} = \frac{1}{Num(y,x)}$, if there is a relationship link from object y to object x, $Num(x, y)$ denotes the number of links from object y to any objects of type X;

  $m_{yx} = 0$, otherwise;
- γ_{YX} denotes the popularity propagation factor of the relationship link from an object of type Y to an object of type X, and $\sum_{\forall Y} \gamma_{YX} = 1$;
- R_{EX}: vector of Web popularity of objects of type X;
- ε: a damping factor which is the probability that the "random object finder" will get bored of following the object relationship links and request another object from the Web using some Web page search engines.

PopRank can be calculated using a simple iterative algorithm.

3. AUTOMATED ASSIGNMENT OF POPULARITY PROPAGATION FACTORS

As we mentioned earlier, in order to calculate the popularity of an object, we need to know the popularity propagation factors of relationship links from the related objects. Because different types of relationships affect the popularity of the related objects in different ways. However it's not practical to manually decide these factors. Since the number of different types of relationships could be very large, and it's hard for a system designer to figure out exactly how much a related object could affect the popularity of another object. However it's easy for the system designer to collect information from domain experts about the popularity orders of some subsets of objects, which are called *partial ranking lists*. In this section we propose an novel approach to automatically learn a set of good popularity propagation factors using these partial ranking lists given by users.

Basically we want to search for a set of popularity propagation factors which could be used in the our *PopRank* model to produce similar rankings for the objects in the partial ranking lists given by domain experts. This becomes a parameter estimation problem. Automated PPF factor assignment is challenging. Since the object relationship graph could be very large, and we may need to wait several hours (or days) to know the quality of an assignment. It would be impractical for us to try thousands of assignments to find a good one. In order to make the learning cost management, we need a good search strategy to find a good assignment by exploring only a small portion of the search space. At the same time, we need to reduce the cost for estimating the quality of each assignment of PPF factors. In the following two subsections, we introduce how we apply existing search algorithms to the PPF estimation problem, and how to reduce the learning cost for each iteration by only considering a subgraph of the entire object relationship graph.

3.1 Search Strategies

The PPF estimation problem is a typical parameter optimization problem, there is a number heuristic based search algorithms that could be adapted to solve the problem. In Figure 3 we show the SAFA (Simulated Annealing for Factor Assignment) algorithm which adapts the simulated annealing algorithm [9] to automatically assign popularity propagation factors.

The basic idea of the algorithms is that we keep examining the neighbors of the current best (or chosen) combination of PPF factors, if a neighbor is better, then it will be chosen as the best combination. We may deliberately choose a worse combination occasionally to avoid being trapped in a local optimal area.

We find the neighbors of the current PPF combination by only changing one factor in a period of time and keep the rest fixed. The neighbors of a single factor $Neighbor(\gamma_{YX})$ is experimentally set as $[\gamma_{YX} - 0.05, \gamma_{YX} + 0.05]$. The cost function $f(\gamma'_{YX})$ is used to measure the quality of the new factor γ'_{YX}. It is computed by replacing the γ_{YX} in the current combination of PPF factors, and applying the new combination of PPF factors to the object relationship graph to compute the *PopRank* score of the training objects. The distance between the ranking results and the ranking given by domain experts is the cost $f(\gamma'_{YX})$ (please see Section

```
Algorithm SAFA(timeout: stopping condition)
  for (each object type $X$)
    $n \leftarrow$ total number of different object types related
        to objects of type $X$;
    for (each related object type $Y$) $\gamma_{YX} \leftarrow \frac{1}{n}$;
  end for
  $t \leftarrow$ a large number;
  do
    for (each object type $X$)
      for (each object type $Y$)
        repeat
          repeat
            randomly select $\gamma'_{YX}$ in $Neighbor(\gamma_{YX})$
            $diff \leftarrow f(\gamma_{YX}) - f(\gamma'_{YX})$;
            if $diff > 0$ then $\gamma_{YX} \leftarrow \gamma'_{YX}$;
            else generate random $x$ in (0,1)
                if $x < exp(-diff/t)$ then $\gamma_{YX} \leftarrow \gamma'_{YX}$;
          until iteration count =
                    $max\_number\_iteration$;
          $t \leftarrow 0.9t$;
        until iteration count =
                    $max\_number\_iteration$;
      end for
    end for
  until timeout;
  return the best combination of $\gamma_{YXs}$;
End SAFA;
```

Figure 3: The SAFA algorithm

5.2 for a detailed discussion on how to measure the distance between two rankings).

3.2 Subgraph Selection

As we mentioned earlier, it would take hours (or days) to estimate the quality of a new PPF factor $f(\gamma'_{YX})$ for large object relationship graph. Since a search algorithm has to try many combinations of PPF factors to find a good one, it would be prohibitively expensive to automatically learn such factors. Fortunately since we only need to know the ranking of the train objects to estimate the quality of a combination of PPF factors, we don't need to use the entire graph during the *PopRank* computation.

We propose to use a subgraph of the entire object relationship graph to reduce the time of ranking the training objects under an assignment of PPF factors. The subgraph consists of a set of concentric circles with the training objects in the center as the core. The circles consist of the objects and their relationship links to other objects in the subgraph. We use the term *$k-$diameter subgraph* to refer the subgraph with k concentric circles which contains all the objects that are less than $k-$links away from the core, and all the links between these objects. We don't need to consider objects that are far away from the core because of the damping factor and their effluence to the core will become smaller and smaller as we increase the distance. The problem now becomes how to find a good subgraph that would not affect the estimation of PPF factors.

We introduce a DiameterEstimator algorithms (see Figure 4) to decide an optimal diameter. We first use the entire

```
Algorithm DiameterEstimator($\delta$:stopping threshold)
    for (each object type $X$)
        $n \leftarrow$ total number of different object types related
            to objects of type $X$;
        for (each related object type $Y$) $\gamma_{YX} \leftarrow \frac{1}{n}$;
    end for
    compute the PopRank scores over the entire graph;
    $R \leftarrow$ the ranking vector of the training objects;
    $R' \leftarrow E$;
    $k \leftarrow 0$;
    while($||R - R'||_1 > \delta$)
        $k++$;
        compute the PopRank scores over the k diameter
            subgraph;
        $R' \leftarrow$ the ranking vector of the training objects;
    end while
    return $k$;
End DiameterEstimator;
```

Figure 4: The DiameterEstimator algorithm

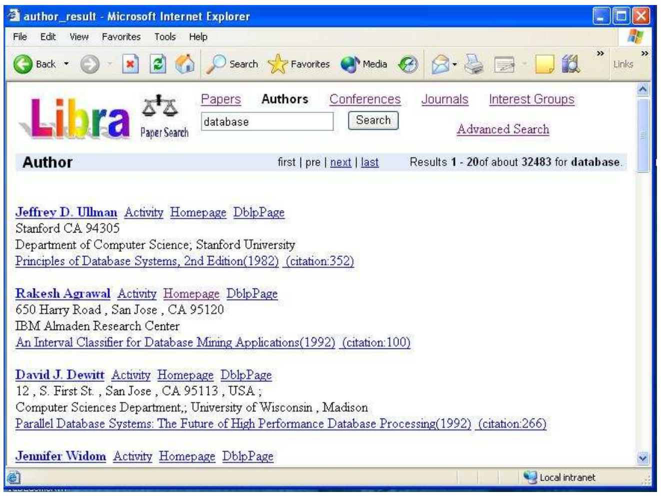
graph and uniform PPF factors to compute the ranking of all the training objects. Then we compute the ranking of all the training objects using the $k-$diameter subgraph. If the difference between the new ranking and the ranking using the entire graph is smaller than the stopping threshold, we consider that the $k-$diameter subgraph is large enough. Since the outside objects of the subgraph will have little impact on estimating the *PopRank* scores of the core objects (i.e. training objects), and their impact may be ignore anyway because of the preset stopping threshold.

4. A CASE STUDY: LIBRA PAPER SEARCH

Based on the *PopRank* model, we have been developing *Libra* [2](see Figure 5), an object-level paper search engine, to help scientists and students locate research materials. *Libra* collects Web information for all types of objects in the research literature including papers, authors, conferences, and journals.

The object information are extracted from Web databases, Web pages, and PDF/PS files. All the information about the same object is integrated as a basic information unit. Currently a paper is uniquely identified by it's title, authors' last names and year information, an author is uniquely identified by his/her full name, and the conferences and journals are identified by their short names (full names are used when no short names are available). The objects are retrieved and ranked according to their relevance to the query and their popularity. The object relevance is calculated using all the collected information about this object, which is stored with respect to each individual attribute. For example, paper information is stored w.r.t. the following attributes: title, author, year, conference, abstract, and full text. In this way, we can also handle structured queries very well, and at the same time, and give different attribute-weights to the hits [5] in different attributes in calculating the relevance score. The details of object relevance score calculation are beyond

[2]http://libra.directtaps.net, username:guest, password:hello libra!


author_result - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media Links
Papers Authors Conferences Journals Interest Groups
Libra Paper Search
database
Search
Advanced Search
Local intranet

Author first | pre | next | last Results **1** - **20**of about **32483** for **database**.

Jeffrey D. Ullman Activity Homepage DblpPage
Stanford CA 94305
Department of Computer Science; Stanford University
Principles of Database Systems, 2nd Edition(1982) (citation:352)

Rakesh Agrawal Activity Homepage DblpPage
650 Harry Road , San Jose , CA 95120
IBM Almaden Research Center
An Interval Classifier for Database Mining Applications(1992) (citation:100)

David J. Dewitt Activity Homepage DblpPage
12 , S. First St. , San Jose , CA 95113 , USA ;
Computer Sciences Department,; University of Wisconsin , Madison
Parallel Database Systems: The Future of High Performance Database Processing(1992) (citation:266)

Jennifer Widom Activity Homepage DblpPage

Figure 5: The Libra User Interface

the scope of this paper.

Compared with traditional paper search engines including *CiteSeer* [1], which search paper information at the document level, *Libra* can retrieve and rank objects such as authors, papers, conferences and journals with respect to a user query. This will greatly benefit junior researchers and students who would be very interested in locating popular scientists, papers, conferences, and journals in their research field. Also, even for recently published papers with few citations, the *PopRank* could assign proper popularity to them if they are published in good conferences or authored by famous researchers.

5. EXPERIMENTS

The *PopRank* model and the PPF estimation algorithms proposed in the paper are fully implemented and evaluated in the context of *Libra*. The goals of the experimental study are: (i) to compare the performance of our *PopRank* model with that of the PageRank model, (ii) to see how different level of subgraph approximation could affect the quality and the efficiency of assigning PPF factors; (iii) to understand the shape of the search space and evaluate the performance of the factor assignment algorithm SAFA. We now describe the datasets and metrics of our experimental evaluation.

5.1 Datasets

Libra currently contains 7 million object relationship links between the collected Web objects including 1 million papers, 650,000 authors, 1700 conferences, and 480 journals. In Figure 1, we show the object relationship graph of these objects. As we can see, there are three different type of relationships links pointing to paper objects: conference/journal $\rightarrow$ paper, author $\rightarrow$ paper, and paper $\rightarrow$ paper. Since the other two types of objects, Conference/Journal objects and Author objects, only has one type of relationship links pointing to them, then the PPF factors for these links is 1. We just need to assign PPF factors for the relationship links pointing to the paper objects.

We have collected 14 partial ranking lists containing ranking information for 67 objects. 8 partial ranking lists (3 for papers, 2 for authors, 2 for conferences, and 1 for journals) containing partial ranking for 45 objects are chosen as training objects and the rest 6 ranking lists (3 for papers, 1 for authors, 1 for conferences, and 1 for journals) containing partial ranking for 22 objects are chosen as test objects. These partial ranking lists are provided by the researchers within the Microsoft Research lab in different areas such as information retrieval, machine learning, databases, computer vision, etc. Here we show a partial ranking list about the ranked conferences in the database community: 1. SIGMOD, 2. VLDB, 3. ICDE, 4. EDBT, 5. ICDT, 6. ER, 7. DEXA, 8. WIDM. We ensure that each object type have at least two ranking lists. A good ranking lists should ensure the correctness of the ranking, and at the same time, the objects close to each other in a ranking list should not have a large gap between their popularity.

5.2 Evaluation Metrics

In order to measure the quality of the ranking results, we compute the distance between two ranking lists of the same set of objects. We need to have an evaluation metrics which not only measures the number of mismatches between these two lists, but also considers the position of these mismatches. For example, if a ranking list switched the ranking of the first object and the second object, the distance between this list and the test list should be greater than that of a ranking list which only switched the last object and the second last object. Since users usually only browse a few top objects in the result list.

We propose the following distance metrics to measure the distance between two ranking lists, R and R'.

$$D(R, R') = \frac{\sum_{i=1}^{n} [(n-i) \times \sum_{j=1 \wedge R'_j \notin \{R_1,\ldots,R_i\}}^{i} 1]}{\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} [(n-i) \times i] + \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} [(n-i) \times (n-i)]}$$

Where n is the total number of objects in the ranking lists, R_i denotes the i^{th} object in ranking list R. Note that the numerator of the formula is used to measure the real distance of these two rankings, and the denominator of the formula is used to normalize the real distance to a number between 0 and 1. As we can see, we give greater penalty to the mismatches for the top objects. For example if the best object is ranked wrongly, the weight for the error will be $n-1$, while if only the second best object is ranked wrongly the weight for the error will be $n-2$.

5.3 Experimental Results

We now present the experimental results.

Understanding the Shape of the Search Space: In Figure 6 we show the search space of the paper-paper relationship PPF factor when the conference-paper relationship PPF Factor is set to 0.3. The data points are collected by uniformly sampling the interval [0, 0.65] of the paper-paper PPF factor with step width 0.005. As we can see that the shape of the search space of the paper-paper PPF is quite predictable if we fix the other factors, there is no dramatic changes within a small interval. Figure 7 shows the search space of the conference-paper relationship PPF factor when the paper-paper PPF is set to 0.3. Although the shape of the search space is not as predictable, there is also no dramatic changes with a small interval. It's clear that a heuristic based search algorithm is suitable for exploring this type of

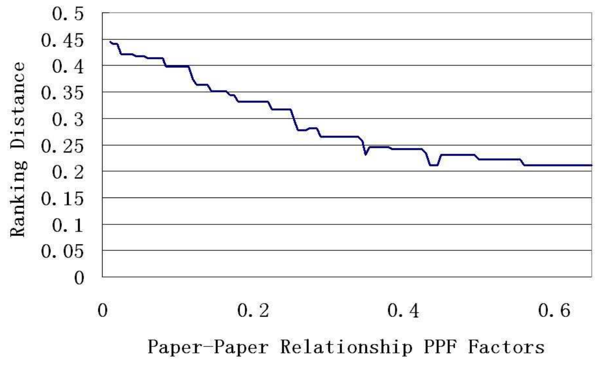


Figure 6: Search Space of Paper-Paper Relationship PPF Factors with Conference-Paper PPF=0.3

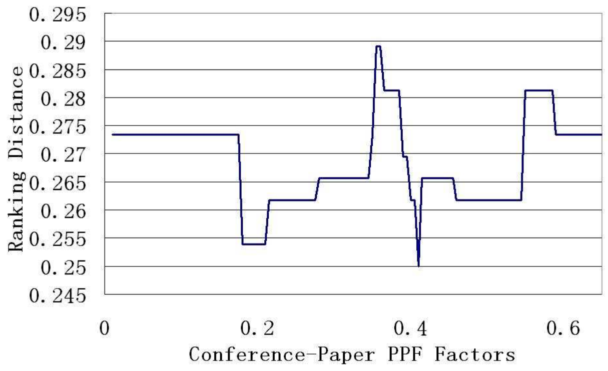


Figure 7: Search Space of Conference-Paper Relationship PPF Factors with Paper-Paper PPF=0.3

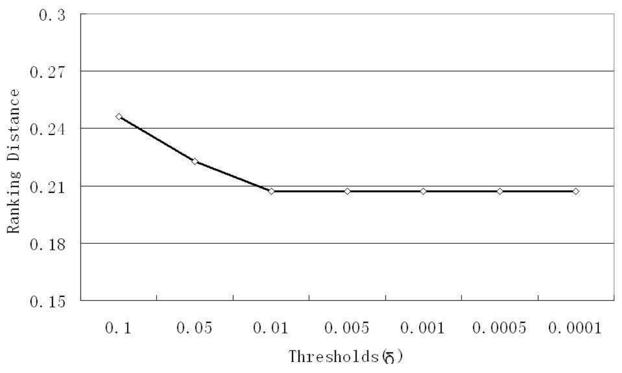


Figure 8: Ranking Distances for Different Thresholds

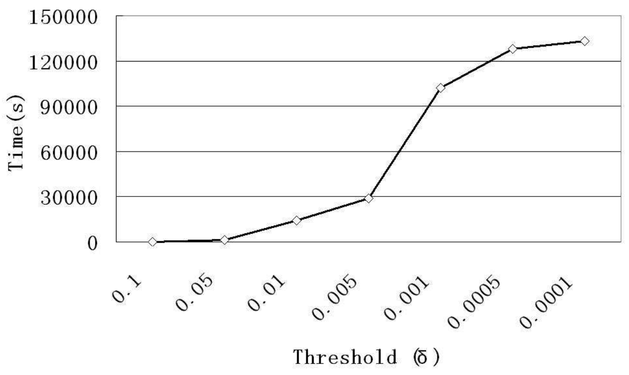


Figure 9: PPF Learning Time for Different Thresholds

search space.

Stop Thresholds: Using the DiameterEstimator algorithm, we automatically select the diameter of the subgraph for PPF factor assignment based on the stopping threshold δ. In Figure 9 and Figure 8, we show the learning time of the PPF factor estimation and the quality of the results using the learned PPF factors for different threshold δ. As we can see the smaller the stopping threshold, the better the quality at the beginning and the longer the learning time. However the quality of the ranking results can not be improved further for thresholds less than 0.01, while PPF factor learning time will keep increasing as the threshold becomes smaller. We can clearly see that 0.01 is a good stop threshold for our datasets.

Subgraph Diameters: In Figure 10, we show how the total number of relationship links change when we increase the diameter of the subgraph. We choose two sets of training objects: 45 core objects and 8 core objects respectively, to study their growing speed of their link graph sizes. As we can see their graph sizes are significantly different when the diameter is smaller than 6.

Based on the previous experimental results, we set the stopping threshold $\delta = 0.01$. The PPF learning time for different diameter sizes using the 45 training objects is shown in Figure 11. From this figure we can found that the learning time could be greatly reduced when we reduce the size of the subgraph.

After we get the learned combination of PPF factors for different diameter sizes, we apply it to the entire graph to calculate the popularity ranking for all objects. We compare the ranking results of the test objects with the ranking from domain experts. Their ranking distance is shown in the figure 12. We can see from the graph, that as we increase the size of the diameter to greater than 4, the ranking distance remains the same, although the PPF factor learning time keeps increasing.

Simulated Annealing: Figure 13 shows the performance of our SAFA algorithm using the 45 training objects. We found that the algorithm tried around 300-400 iterations to find out the optimal PPF assignment. It maybe a local optimal solution, however we tried for another 1100 iterations and found no improvement.

PopRank versus PageRank: For comparison, we implemented the PageRank algorithm which assigns a uniform popularity propagation factor for all links. Figure 14 shows the comparison of our *PopRank* algorithm with the PageRank algorithm in several test datasets. In order to show that our *PopRank* model works well even with a small number of training objects, we use the previous 22 test objects as training objects here, and 6 subsets of the previous 45 training objects are selected as the test datasets here. The number of objects in a test dataset changes from 45 to 20. The results show that our algorithm is significantly better than

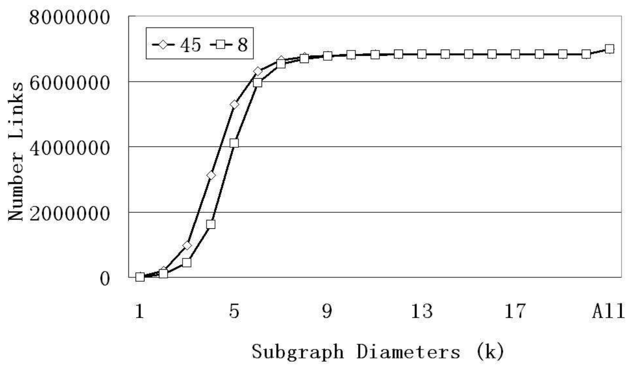


Figure 10: Number of Object Relationship Links for Different Subgraph Diameters

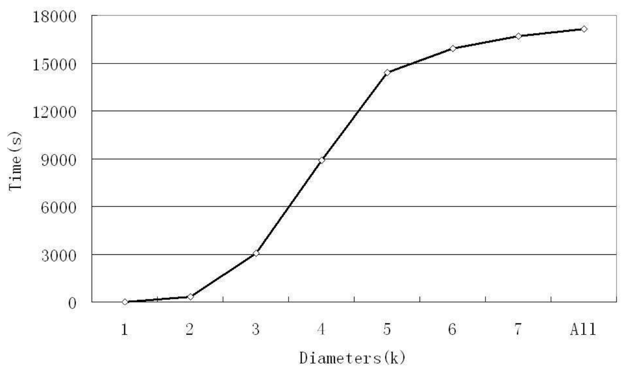


Figure 11: PPF Factor Learning Time for Different Subgraph

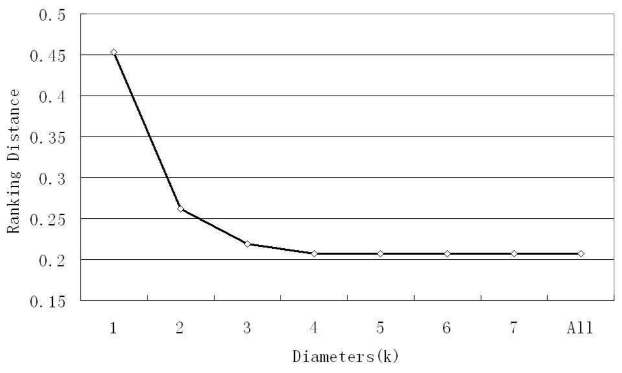


Figure 12: Ranking Distance for Different Diameters

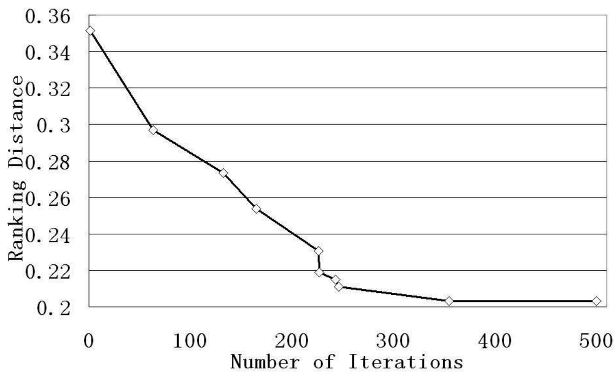


Figure 13: Simulated Annealing

PageRank for all the test datasets. In average, our ranking accuracy increases about 50%, Which clearly illustrates that naively applying PageRank on object relationship graph is essentially not feasible.

Number of Training Objects: In Figure 15, we observe how different number of training objects affects the quality of ranking results. As we can see when we increase the number of training objects, the ranking accuracy keeps improving. Currently, we only collected 67 ranked objects for training and test. In the future, we plan to include more objects for training. Hopefully, the accuracy will be further improved.

6. RELATED WORK

Brin and Page first introduce the PageRank technique [13] to calculate the importance of a Web page based on the scores of the pages pointing to the page. Hence, Web pages pointed by many high quality pages become important as well. Alternatively, the importance score of a Web page is equal to the probability that a random surfer, starting from a random page, will reach the Web page at a specific time. Since the PageRank model considers that all the links have the same authority propagation factors, it could not be directly applied to our object-level ranking problem.

The PageRank model has also been adapted to structure databases. Guo et al. [7] introduce XRANK to rank XML elements using the link structure of the database. Balmin et al. propose the ObjectRank system [4] which applies the random walk model to keyword search in databases modelled as labelled graphs. A similar notion of our popularity propagation factors called authority transfer rates is introduced. In their relevance feedback survey study, the authors find out that using different transfer rates for different edge types is effective. However the papers did not discuss how these authority transfer rates could be assigned.

Xi et al. [17] propose a unified link analysis framework called "link fusion" to consider both the inter- and intra-type link structure among multi-type inter-related datan objects. The PageRank and HITS algorithms [10] are shown to be special cases of the unified link analysis framework. Although the paper mentioned some similar notion of our popularity propagation factor, however how to assign these factors is considered as the most important future research work in the paper. Furthermore, our *PopRank* model itself is also significantly different from the link fusion framework. In our *PopRank* model we take both the Web popularity of an object and the popularity from the object relationship graph into account. Combining both types of popularity is important, especially for application domains where objects are widely available on Web databases and Web pages. For example, if we want to build a product search engine to rank the product related objects, the Web popularity of these objects could be very useful to calculate the popularity of these products, only using the object relationship graph

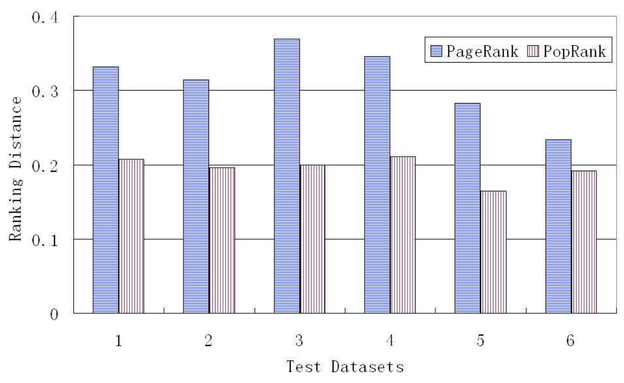


Figure 14: PopRank versus PageRank

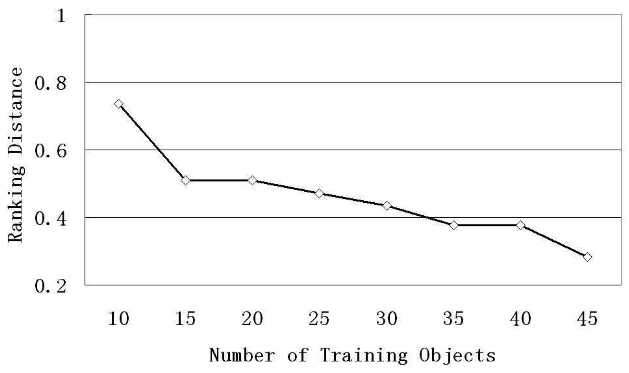


Figure 15: Ranking Distances for Different Number of Training Objects

could lead to unreasonable ranking.

7. CONCLUSION

This paper studies how to calculate the object popularity scores of Web objects based on their Web popularity and the object relationship graph. Traditional PageRank algorithms is no longer valid because of the existence of heterogeneous relationships between objects. We propose to automatically assign a popularity propagation factor for each type of object relationship. Specifically the contributions of the paper are:

(i) A *PopRank* model which considers both the Web popularity of an object and the object relationship graph to calculate the *PopRank* score of the Web object;

(ii) An automated approach for assigning popularity propagation factors for different types of object relationships using partial ranking lists from domain experts. We propose to use a subgraph of the entire object relationship graph to efficiently search for good propagation factors;

(iii) The experiments are done in the context of *Libra*, an object-level Web search engine indexing 1 million papers.

8. REFERENCES

[1] Citeseer. Scientific Literature Digital Library. http://citeseer.ist.psu.edu.

[2] Libra. Object-level Paper Search Engine. http://libra.directtaps.net, username:guest, password:hello libra!

[3] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. In *Proc. Workshop on Management of Semistructured Data*, Tucson, 1997.

[4] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. Authority-based keyword queries in databases using objectrank. In *Very Large Data Bases (VLDB)*, 2004.

[5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[6] Deng Cai, Xiaofei He, Ji-Rong Wen, and Wei-Ying Ma. Block-level link analysis. In *ACM SIGIR Conference (SIGIR)*, 2004.

[7] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *ACM SIGMOD*, 2003.

[8] Bin He, Kevin Chen chuan Chang, and Jiawei Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *Knowledge Discovery and Data Mining (KDD)*, 2004.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[10] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5), 1999.

[11] Nickolas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, 1997.

[12] Bing Liu, Robert Grossman, and Yanhong Zhai. Mining data records in web pages. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries, 1998.

[14] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *World Wide Web conference (WWW)*, 2004.

[15] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Knowledge Discovery and Data Mining (KDD)*, 2002.

[16] Jiying Wang, Ji-Rong Wen, Frederick H. Lochovsky, and Wei-Ying Ma. Instance-based schema matching for web databases by domain-specific query probing. In *Very Large Data Bases (VLDB)*, 2004.

[17] Wensi Xi, Benyu Zhang, Yizhou Lu, Zheng Chen, Shuicheng Yan, Huajun Zeng, Wei-Ying Ma, and Edward A. Fox. Link fusion: A unified link analysis framework for multi-type interrelated data objects. In *WWW*, 2004.