UNIVERSITY OF CALIFORNIA, SAN DIEGO

Dynamic Link-based Ranking over Large-Scale Graph-Structured Data

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Heasoo Hwang

Committee in charge:

Professor Yannis Papakonstantinou, Chair
Professor Richard Belew
Professor Alin Deutsch
Professor Gert Lanckriet
Professor Victor Vianu

2010

The dissertation of Heasoo Hwang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2010

DEDICATION

To my family and friends,

especially to my parents in Korea.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I thank my advisor Yannis Papakonstantinou for sharing valuable insights, guiding my research, and showing endurance throughout my Ph.D. years. I know that without him, nothing could have been done. Also, I would like to thank the other professors in our lab, Alin Deutsch and Victor Vianu, for providing me of warm encouragement and invaluable advices when I really needed them. I also thank my other committee members, Richard K. Belew and Gert Lanckriet, for their time and consideration.

I thank my former and current lab colleagues, too. Especially, I thank my cohorts at UCSD database group, Yannis Katsis and Emiran Curtmola, for providing me of insightful comments and moral support all the time. Also, I offer my sincere gratitude to my excellent mentors and collaborators, Andrey Balmin, Vagelis Hristidis, and Alex Ntoulas. I truly think it was luck and privilege for me to work with them. I also thank many great collaborators I had at UC San Diego, IBM Almaden Research Center, and Microsoft Search Labs in San Jose. They include Amarnath Gupta(SDSC), Berthold Reinwald(IBM Almaden), Hamid Pirahesh(IBM Almaden), Erik Nijkamp(IBM Germany), Yannis Sismanis(IBM Almaden), Frank van Ham(IBM Watson), Hady Lauw(Microsoft Research), and Lise Getoor(University of Maryland).

My special thanks go to my family, Kyung Ki Hwang, Ae Kyung Park, Sang Rae Shim, Hyunho Hwang and Hyunchul Hwang, for their love, prayers, and so many phone calls. Also, I would like to thank my best friends, Ji Hee Jung, Ki Yeon Kim, and YouRyang Seo, for their firm presence and strong support during my Ph.D. studies. I also thank people in San Diego Onnuri Church for sharing so many good times with me and cheering me up. Last but not least, I would like to express my immeasurable gratitude to my God and Jesus Christ for saving me, caring me, loving me, and walking with me throughout my life and forever.

Parts of Chapter 2 were published in Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD-2006), pp 796-798 and ACM Transactions on Database Systems 2008, 33(1) (TODS-2008). Heasoo Hwang, Vagelis Hristidis, and Yannis Papakonstantinou, "ObjectRank: a system for authority-based search on databases" and Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou, "Authority-based keyword search in databases". The dissertation author and Vagelis Hristidis were the primary investigators and authors of these papers.

Chapter 3 was published in Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE-2009) and will appear in IEEE Transactions on Knowledge and Data Engineering 2010, Special Issue on the Best Papers of ICDE09 (TKDE-2010). Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp, "BinRank: Scaling Dynamic Authority-Based Search Using Materialized SubGraphs" and its extended version. The dissertation author was the primary investigator and author of these papers.

Chapter 4 was published in Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD-2006), pp 796-798 and ACM Transactions on Database Systems 2008, 33(1) (TODS-2008). Heasoo Hwang, Vagelis Hristidis, and Yannis Papakonstantinou, "ObjectRank: a system for authority-based search on databases" and Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou, "Authority-based keyword search in databases". The dissertation author and Vagelis Hristidis were the primary investigators and authors of these papers.

VITA AND PUBLICATIONS

2000	Bachelor of Science in Computer Science, Seoul National University, Seoul, Korea
2002	Master of Science in Computer Science, Seoul National University, Seoul, Korea
2010	Doctor of Philosophy in Computer Science, University of California, San Diego

Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp, "BinRank: Scaling Dynamic Authority-Based Search Using Materialized SubGraphs"(extended version), *To appear in IEEE Transactions on Knowledge and Data Engineering (TKDE), Speical Issue on the Best Papers of ICDE 2009*, 2010.

Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp, "BinRank: Scaling Dynamic Authority-Based Search Using Materialized SubGraphs", *In Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE-2009)*, pp. 66-77, 2009.

Akanksha Baid, Andrey Balmin, Heasoo Hwang, Erik Nijkamp, Jun Rao, Berthold Reinwald, Alkis Simitsis, Yannis Sismanis, and Frank van Ham, "DBPubs: Multidimensional Exploration of Database Publications", *In Proceedings of the 34th International Conference on Very Large Data Bases (VLDB-2008)*, pp. 1456-1459, Auckland, New Zealand, August 24-30, 2008.

Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou, "Authority-based keyword search in databases", *ACM Transactions on Database Systems (TODS)*, 33(1), 2008.

Heasoo Hwang, Andrey Balmin, Hamid Pirahesh, and Berthold Reinwald, "Information discovery in loosely integrated data", *In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD-2007)*, pp. 1147 - 1149, 2007.

Heasoo Hwang, Andrey Balmin, Hamid Pirahesh, and Berthold Reinwald, "ObjectRank: a system for authority-based search on databases", *In Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD-2006)*, pp. 796-798, 2006.

ABSTRACT OF THE DISSERTATION

Dynamic Link-based Ranking over Large-Scale Graph-Structured Data

by

Heasoo Hwang

Doctor of Philosophy in Computer Science

University of California, San Diego, 2010

Professor Yannis Papakonstantinou, Chair

Information Retrieval techniques have been the primary means of keyword search in document collections. However, as the amount and the diversity of available semantic connections between objects increase, link-based ranking methods including ObjectRank have been proposed to provide high-recall semantic keyword search over graph-structured data. Since a wide variety of data sources can be modeled as data graphs, supporting keyword search over graph-structured data greatly improves the usability of such data sources. However, it is challenging in both online performance and result quality.

We first address the performance issue of dynamic authority-based ranking methods such as personalized PageRank and ObjectRank. Since they dynamically rank nodes in a data graph using an expensive matrix-multiplication method, the online execution time rapidly increases as the size of data graph grows. Over the English Wikipedia dataset of 2007, ObjectRank spends 20-40 seconds to compute query-specific relevance scores, which is unacceptable. We introduce a novel approach, BinRank, that approximates dynamic link-based ranking scores efficiently. BinRank partitions a dictionary into bins of relevant keywords and then constructs materialized subgraphs(MSGs) per bin in preprocessing stage. In query time, to produce highly accurate top-K results efficiently, BinRank uses the MSG corresponding to the given keyword, instead of the original data graph.

PageRank and ObjectRank calculate the global importance score and the query-specific authority score of each node respectively by exploiting the link structure of a given data graph. However, both measures favor nodes with high in-degree that may contain popular yet generic content, and thus those nodes are frequently included in top-K lists, regardless of given query. We propose a novel ranking measure, Inverse ObjectRank, which measures the content-specificity of each node by traversing the semantic links in the data graph in the reverse direction. Then, we allow users to adjust the importance of the three ranking measures (global importance, query-relevance, and content-specificity) to improve the quality of search results.

Chapter 1

Introduction

Recently, the variety and the size of digital information sources have grown very rapidly. In particular, there is an increasing number of data sources that contain not only *data objects* but the *links* that connect semantically relevant objects have become widely prevalent. In this dissertation, we refer to such sources as graph-structured data sources. We target on improving their usefulness to end-users by providing efficient and effective search functionality, based on the fact that the semantic link information cannot be exploited easily by ordinary users, but is very helpful in finding relevant data objects effectively.

Bibliographic datasets such as the DBLP database[1] and the Citeseer dump data[2] are good examples of graph-structured data sources. They provide relational tuples or XML elements that correspond to typed data objects such as authors and publications, as well as typed semantic links such as authorships and citations represented as foreign-key relationships or XPointers/XLinks connecting authors and publications. As stated in [BHP04], such semantic links are very useful in improving the quality of search results, especially when the importance of a data object is transferred to neighboring objects via semantic links connecting them. For instance, if there is a set of papers, P, that are important in the "OLAP" research area, and all of the papers cite paper p_{cited}, we can intuitively see that p_{cited} is also an important paper. This can be explained by using the authority

[1]http://www.informatik.uni-trier.de/ ley/db/
[2]http://citeseer.ist.psu.edu/

Table 1.1: Examples of Graph-Structured Data in Various Domains

D_1:	Publications dataset in a relational database (Structured database)
D_2:	Wikipedia articles in an XML database (Semi-structured database)
D_3:	A collection of social networking web pages including profile pages, where the profile of each person is connected to those of co-workers, universities and companies (A collection of XML documents)

Table 1.2: Examples of Keyword Queries over Graph-Structured Data in Table 1.1

Q_1:	Who are important researchers (or papers) in the *"OLAP"* research?
Q_2:	Find Wikipedia articles highly relevant to *"PageRank"*.
Q_3:	Which universities are closely related to *"CompanyX"*?

(importance) transfer mechanism: the importance scores of the papers in P flow into p_{cited} through the citation links from P to p_{cited}. Therefore, our target graph-structured data sources are the ones on which the authority transfer mechanism is observed.

As many ordinary users want to access and use graph-structured data sources such as the datasets in Table 1.1, supporting an efficient and effective keyword search functionality over graph-structured data has become increasingly important. Text-based ranking methods including traditional Information Retrieval (IR) techniques have been one of the most commonly used approaches to enable keyword search over an unstructured collection of documents. Given a keyword query, they usually define the degree of relevance of each document to the query by using the number of occurrences of the given keywords in the textual content of the document. However, the IR techniques that rely only on keyword occurrences are not effective in generating semantically meaningful search results, especially when the containment/absence of given keywords and the relevance to the query do not coincide. For instance, if there are objects whose textual contents do not contain the given keywords but are highly relevant to the query, traditional IR techniques cannot find them. Also, if two documents with the same document length have the same number of keyword occurrences, text-based ranking methods rank them as equally relevant to the query, which is not always the case. In this way, traditional IR techniques may suffer from problems regarding the quality of search results such as low recall search and semantically incorrect ranking. This is due to the fact that the relevance information is not fully expressed in the textual contents of objects, but often hidden in the link structure between objects.

Latent Semantic Indexing (LSI) [DDF^{+}90] is a relatively new IR technique that exploits the latent semantic structure of the pattern of keyword usage across documents to generate an expanded set of results with better recall. By examining keyword co-occurrences across a collection of documents, it captures latent semantic associations of keywords. Thus, it can find relevant documents that do not contain the given keywords if they contain keywords semantically related to the given keywords. Even though it partially overcomes the deficiencies of assuming independence of keywords such as the low recall problem of traditional IR techniques we discussed above, it does not provide a solid solution for the second problem described above. This is because it still relies only on the textual contents of documents, and does not exploit the relevance information hidden in the link structure.

Dynamic link-based ranking methods such as HITS [Kle99], topic-sensitive PageRank [Hav02], personalized PageRank [JW03], and ObjectRank [BHP04] proposed to measure the relevance of objects to a given keyword query effectively by exploiting the semantic link information between objects. ObjectRank [BHP04], in particular, is a variation of personalized PageRank [JW03] that performs dynamic link-based ranking for high-recall semantic keyword search over graph-structured data. For example, when a user query Q_1 in Table 1.2 is given, ObjectRank can find not only the authors of papers containing the keyword "OLAP" in their titles, but the researchers whose are very important in the "OLAP" research community even though their paper titles do not contain "OLAP". Those researchers are highly relevant to the "OLAP" research since their papers are frequently 'cited by' many papers related to "OLAP", or they have been closely collaborating with authors related to "OLAP". In fact, as demonstrated in previous works [Kle99, Hav02, JW03, BHP04], by leveraging semantic link information between data objects, the quality of search results over graph-structured data can be greatly improved.

In this dissertation, we aim to overcome the deficiencies of text-based search methods we discussed above by leveraging the relevance information hidden behind the link structure of graph-structured data. In particular, we focus on generating semantically meaningful search results for top-K keyword queries in a more efficient and effective way. To provide high-recall semantic search results for keyword queries (e.g. Q_1-Q_3 in Table 1.2) over graph-structured data (e.g. D_1-D_3 in Table 1.1), we follow the ranking method proposed by Balmin et al., ObjectRank [BHP04], that applied the dynamic link-based search technique of personalized PageRank [JW03] to implement an effective keyword search functionality over graph-structure data.

ObjectRank [BHP04] is an effective top-K keyword search system that produces high-quality search results by leveraging both the textual contents of data objects and the link structure of graph-structured data: for a given keyword query, ObjectRank uses IR techniques to identify a set of objects whose textual contents are related to the given keywords(in [BHP04], Balmin et al. used an IR function checking keyword containment). The query-specific importance scores of these objects are initialized to nonzero values, while relevance scores of the rest of the objects are set to 0. Then, by exploiting the link structure of graph-structured data, ObjectRank computes query-specific authority scores of objects, and finds top-K most relevant objects with respect to the given keyword query.

It is worth noting that the efficient and effective link-based search framework we propose in this dissertation is not limited to support keyword search using ObjectRank, but can be applied to implement various ranking functionalities based on the personalized PageRank. Notice that this dissertation addresses both the performance and the semantic issues of ObjectRank [BHP04], a variant of personalized PageRank. The background of link-based search methods including ObjectRank [BHP04] is given in Chapter 2.

In the following sections, we briefly describe the problems we solve and give overviews of the solutions we suggest in this thesis, *BinRank* [HBRN09, HBRN10] and *Inverse ObjectRank* [HHP06, HHP08], to support a more efficient and effective link-based keyword search functionality over graph-structured data.

1.1 Efficiency of Dynamic Link-based Search

Unlike the original PageRank [PBMW98] that measures global importance scores of nodes in the Web graph offline, ObjectRank [BHP04] is a dynamic link-based search method in that it calculates query-specific importance scores for each given keyword query. Although link-based search methods such as topic-sensitive PageRank and personalized PageRank in [Hav02, JW03] also generate query-specific rankings, the degree of personalization is limited to 16 topics or a subset of hub nodes in a human-constructed directory. ObjectRank, by contrast, supports full personalization in order to cover all the keyword queries in a dictionary, and generates high-recall semantic search results for a given keyword query.

Despite the search effectiveness of ObjectRank, however, it is often infeasible to implement keyword search functionality over a graph-structured data source using ObjectRank as it is because of the performance issue of ObjectRank. At high level, for a given query and a data graph, ObjectRank uses iterative matrix multiplication method to rank the nodes in the graph with respect to the query in query time. To this extent, as the size of a data graph becomes larger, the matrix multiplication takes more time to reach the fixpoint, sometimes resulting in prohibitively long query time. Alternatively, building an index of pre-computed results for some or all keywords is proposed in [BHP04], yet it is not a feasible solution either because of its expensive preprocessing cost.

Let us consider a collection of English Wikipedia articles, where various types of semantic links such as 'definition' links, 'see also' links, and 'category' links through which authority flows between Wikipedia[3] articles. Let us denote the data graph modeling English Wikipedia dataset as $G_{wiki} = (V_{wiki}, E_{wiki})$. Since G_{wiki} contains 3.2 million nodes and 109 million links, even a fully optimized in-

[3]http://en.wikipedia.org

memory implementation of ObjectRank takes 20-50 seconds to rank nodes in the graph. In the off-line mode, ObjectRank precomputes top-k results for a query workload in advance. This precomputation is very expensive and requires a lot of storage space for precomputed results. Moreover, this approach is not feasible for all keywords outside the query workload that a user may search for, i.e. for all keywords in the dataset dictionary. For example, on the same Wikipedia dataset, the full dictionary precomputation would take about a CPU-year.

As we observed in the experimental results in Figure 3.13, the running time of link-based keyword search is almost linear in the size of a data graph (the number of links). In fact, the Pearson's correlation coefficient between them is estimated to 0.938 using high frequency keywords, which indicates a very strong correlation. We noticed that users are interested in top-K search lists with a small K such as 20 or 100, while the data graph modeling a real-life dataset such as G_{wiki} is huge in size. Interestingly, since G_{wiki} covers many other topics such as historical events or movie actors, the major portion of G_{wiki} is irrelevant to any given keyword query. For instance, the number of Wikipedia articles irrelevant to query "OLAP" exceeds 3 million, which counts for more than 90% of $|V_{wiki}|$. We observed that ObjectRank returns the same top-K search list for "OLAP" even though we run ObjectRank in the subgraph of G_{wiki} induced from relevant nodes only, not in the original data graph G_{wiki}. This is because the subgraph sufficiently captures the search context of the given query.

We introduce *BinRank* [HBRN09, HBRN10], a system that approximates ObjectRank results by utilizing a hybrid approach inspired by materialized views in traditional query processing. We materialize a number of relatively small subsets of the data graph in such a way that any keyword query can be answered by running ObjectRank on only one of the subgraphs. The subgraphs BinRank generates satisfy the following properties. Firstly, in order to improve the running time of ObjectRank, the subgraphs should be small. Secondly, each subgraph should well capture the search context of the corresponding keyword queries. Notice that we aim to speed up the link-based keyword search process, while producing the same top-K search lists ObjectRank generates in the entire data graph. Lastly, to provide full coverage of the dictionary, every keyword query can be answered using the subgraphs.

BinRank generates the subgraphs by partitioning all the terms in the corpus based on their co-occurrence, executing ObjectRank for each partition using the terms to generate a set of random walk starting points, and keeping only those objects that receive non-negligible scores. The intuition is that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms.

We demonstrate that BinRank can achieve sub-second query execution time on the English Wikipedia dataset, while producing high-quality search results that closely approximate the results of ObjectRank on the original graph. The Wikipedia link graph contains about 10^8 edges, which is at least two orders of magnitude larger than what prior state of the art dynamic authority-based search systems have been able to demonstrate. Our experimental evaluation investigates the trade-off between query execution time, quality of the results, and storage requirements of BinRank.

1.2 Measuring Authority and Specificity

Another major contribution of this dissertation is to propose and evaluate a novel ranking measure, *Inverse ObjectRank*, that captures the degree of specificity of each data object in a given data graph with respect to a given keyword query. Various approaches including text-based methods and link-based methods have already been suggested to produce effective top-K keyword search results. In this dissertation, we particularly focus on the ranking techniques measuring various important aspects of data objects by leveraging the links connecting relevant data objects.

Text analysis methods such as traditional information retrieval techniques focus on measuring the relevance between each data object and the keyword query by considering the textual contents of data objects. However, such methods are incapable of identifying relevant data objects if their textual contents do not contain the given keyword. Motivated by the observation that the relevance information is often hidden behind the connections between data objects, Balmin et al.[BHP04] proposed a link-based ranking measure for relevance, ObjectRank, that leverages the link structure modeling connections between relevant data objects. Based upon the proven effectiveness of link-based ranking measures, we propose a novel ranking measure for specificity to capture another important aspect of data objects by exploiting the link structure.

The ObjectRank system [BHP04] assumes that users prefer the data objects that are highly relevant to the given query and very popular in general. It captures these two important features of data objects, the relevance to a given keyword query and the global importance, by analyzing the link structure for a given query. First, the *relevance* measure originally suggested by Balmin et al. [BHP04] captures the query-specific authority score, i.e., the degree of relevance to a given keyword query, which is referred to as the *ObjectRank* score. In addition to the query-specific importance, Balmin et al. [BHP04] applied the PageRank algorithm [PBMW98] to the given data graph as described in Section 2.2 to measure the query-independent importance, and obtained the *global ObjectRank* score of each data object. These link-based measures successfully capture the relevance and the global importance of each data object, enabling the ObjectRank system to generate high-recall, semantic keyword search results.

However, we frequently observed that popular data objects with generic contents such as database textbooks appear in the top-K result lists generated by the ObjectRank system regardless of given keyword queries. Notice that both ranking measures, ObjectRank and global ObjectRank, of the ObjectRank system focus on capturing the authority of each data object in the data graph, but cannot distinguish between the data objects specifically relevant to the given query(*'authoritative-specific'*) and the ones with generic contents(*'authoritative-non-specific'*). In this dissertation, we introduce a novel link-based specificity measure, *Inverse ObjectRank*, and claim that we can improve the effectiveness of top-K keyword search by considering both the authority and the specificity of each data object in the data graph. Our specificity measure allows users to penalize a data object for its content-specificity or topical diversity.

In the following usage scenario, we provide two situations with different user preferences, where authoritative-non-specific objects and authoritative-specific objects are preferred, respectively. Through this, we highlight our motivation for the specificity measure that enables users to retrieve more authoritative-specific objects in the top-K search lists:

> *Let us consider a professor who needs to give a reading list to a first year graduate student who starts research on a topic, say "XML database storage". Being a first year student, he/she likely has no background knowledge on database issues pertaining to XML and semistructured data in general. In this case, you may want to provide an authoritative papers list where it is OK (indeed desirable) to include a few seminal papers on XML and semistructured databases, even though they may not be related to storage in particular. Such seminal papers are a good starting point for the student. These papers are authoritative-non-specific papers.*
> *Now assume that the professor wants to produce a reading list for someone who already knows the basics of XML databases and of conventional (relational) storage systems. He now cares about the specific papers in XML storage, in particular. Such papers are authoritative-specific papers*

In Chapter 4, we describe our new ranking measure, *Inverse ObjectRank* [HHP06, HHP08], which exploits the link structure in the reverse direction to measure how specific each data object is with respect to a given query. When there are two nodes with similarly high authority scores, Inverse ObjectRank enables users to distinguish the nodes whose content is specifically relevant to the given query from authoritative nodes with generic content. In this way, by combining two existing authority measures, ObjectRank and global ObjectRank, with our new specificity measure, Inverse ObjectRank, we can further improve the effectiveness of link-based top-K keyword search. Users may obtain search results of better quality by combining these three link-based ranking measures according to their requirements. We conducted user surveys with domain experts to assess the quality of search results produced by various combinations of these measures.

1.3 Dissertation Overview

In this dissertation, we focus on graph-structured data sources in which there exists authority flow between objects through links. For those data sources, we want to provide a more efficient and effective authority-based search system. In particular, we propose our solutions to both the performance and semantic issues of dynamic authority-based search methods such as ObjectRank [BHP04] and personalized PageRank [Hav02, JW03].

We first review the background pertaining to this dissertation in Chapter 2. Just like the Web that can be viewed as a graph with authority flow, various data sources such as the DBLP database and the collection of Wikipedia articles can also be modeled as graphs. ObjectRank [BHP04] applied the idea of authority-based search suggested by personalized PageRank (see Section 2.1) for keyword search over databases. ObjectRank modeled a database as a labeled graph that captured meaningful authority flows via various types of semantic links between objects in the database. Throughout Chapter 2, we review the data model and the query processing stage of ObjectRank, and then discuss about the quality superiority of ObjectRank search results. To enable such authority-based search over real-life datasets, however, the performance of ObjectRank (and personalized ObjectRank) should be improved.

Therefore, in Chapter 3, we address the performance issue of dynamic authority-based search methods such as ObjectRank [BHP04]. Since ObjectRank calculates query-specific relevance scores in runtime using expensive iterative matrix multiplication, the ObjectRank execution time rapidly increases as the size of data graph grows. In fact, the same problem appears in all the dynamic authority-based ranking methods including personalized PageRank [Hav02, JW03]. To make dynamic authority-based keyword search feasible over large-scale graph-structured data, we suggest *BinRank*, a system that approximates ObjectRank results by utilizing a hybrid approach inspired by materialized views in traditional query processing. Our experimental evaluation investigates the trade-off between query execution time, quality of the results, and storage requirements of BinRank.

In Chapter 4, we focus on the semantic quality issue of link-based keyword search. In particular, we suggest *Inverse ObjectRank*, a novel specificity measure based on link structure, and combine it with various types of existing link-based ranking measures such as the global importance(PageRank) and the query-specific relevance(ObjectRank). We provide a demo system that allows users to adjust the importance of these various link-based ranking measures. We analyze and evaluate the semantic contribution of Inverse ObjectRank and compare various combining functions by assessing the quality of search results through experiments.

In Chapter 5, we summarize and give conclusions.

Chapter 2

Background - ObjectRank

2.1 Authority-based Search

PageRank We first describe the essentials of PageRank and authority-based search, and the random surfer intuition. Let (V, E) be a graph, with a set of nodes $V = \{v_1, \ldots, v_n\}$ and a set of edges E. A surfer starts from a random node (web page) v_i of V and at each step, he/she follows a hyperlink with probability d or gets bored and jumps to a random node with probability $1 - d$. The PageRank value of v_i is the probability $r(v_i)$ that at a given point in time, the surfer is at v_i. If we denote by $\mathbf{r}$ the vector $[r(v_1), \ldots, r_(v_i), \ldots, r(v_n)]^T$ then we have

$$\mathbf{r} = d\mathbf{A}\mathbf{r} + \frac{(1-d)}{|V|}\mathbf{e} \tag{2.1}$$

where $\mathbf{A}$ is a $n \times n$ matrix with $A_{ij} = \frac{1}{OutDeg(v_j)}$ if there is an edge $v_j \to v_i$ in E and 0 otherwise, where $OutDeg(v_j)$ is the outgoing degree of node v_j. Also, $\mathbf{e} = [1, \ldots, 1]^T$.

Personalized PageRank with Base Set The above PageRank equation is typically precomputed before the queries arrive and provides a global, keyword-independent ranking of the pages. Instead of using the whole set of nodes V as the *base set*, i.e., the set of nodes where the surfer jumps when bored, one can use an arbitrary subset S of nodes, hence increasing the authority associated with the nodes of S and the ones most closely associated with them. In particular, a *base vector* $\mathbf{s} = [s_0, \ldots, s_i, \ldots, s_n]^T$ can be defined, where s_i is 1 if $v_i \in S$ and 0 otherwise. The PageRank equation is then

$$\mathbf{r} = d\mathbf{A}\mathbf{r} + \frac{(1-d)}{|S|}\mathbf{s} \tag{2.2}$$

Regardless of whether one uses Equation 2.1 or Equation 2.2 the PageRank algorithm solves this fixpoint using a simple iterative method, where the values of the (k+1)-th execution are calculated as follows:

$$\mathbf{r}^{(k+1)} = d\mathbf{A}\mathbf{r}^{(k)} + \frac{(1-d)}{|S|}\mathbf{s} \tag{2.3}$$

The notion of the base set S was suggested in [BP98] as a way to do personalized rankings, by setting S to be the set of bookmarks of a user. In [Hav02], it was used to perform topic-specific PageRank on the Web. ObjectRank [BHP04] takes it one step further and uses the base set to estimate the relevance of a node to a keyword query. In particular, the base set of ObjectRank consists of the nodes that contain the keyword as explained in Section 2.2.

Convergence of PageRank The computation of Equation (2.2) terminates when $\mathbf{r}$ converges, which is guaranteed to happen under very common conditions [MR95]. In particular, the authority flow graph needs to be irreducible (i.e., (V, E) be strongly connected) and aperiodic. The former is true due to the damping factor d, while the latter happens in practice.

2.2 Overview of ObjectRank

PageRank [BP98] is an excellent tool to rank the global importance of the pages of the Web. However, PageRank measures the global importance of the pages, independently of a keyword query. More recent works [Hav02, RD02] apply PageRank to estimate the relevance of pages to a keyword query. ObjectRank [BHP04] appropriately extends and modifies PageRank to per-

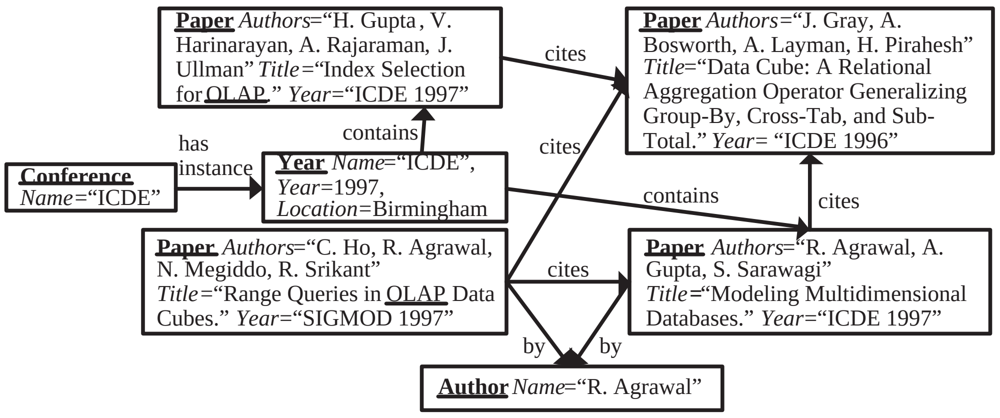

Paper Authors="H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman" Title="Index Selection for OLAP." Year="ICDE 1997"
Paper Authors="J. Gray, A. Bosworth, A. Layman, H. Pirahesh" Title="Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total." Year= "ICDE 1996"
cites
contains
has instance
cites
Conference Name="ICDE"
Year Name="ICDE", Year=1997, Location=Birmingham
contains
cites
Paper Authors="C. Ho, R. Agrawal, N. Megiddo, R. Srikant" Title="Range Queries in OLAP Data Cubes." Year="SIGMOD 1997"
cites
Paper Authors="R. Agrawal, A. Gupta, S. Sarawagi" Title="Modeling Multidimensional Databases." Year="ICDE 1997"
by
by
Author Name="R. Agrawal"

Figure 2.1: A subset of the DBLP graph

form keyword search in databases for which there is a natural flow of authority between their objects (e.g., bibliographic or complaints databases).

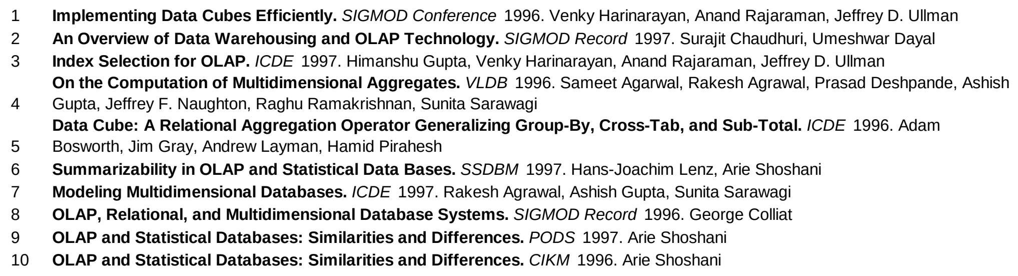
Given a keyword query, ObjectRank ranks the results according to two factors: the relevance to the query and the global importance of the result. Both factors are handled using authority-flow techniques that exploit the link structure of the data graph, in contrast to traditional Information Retrieval. The relevance is computed using the ObjectRank metric which is a keyword-specific adaptation of PageRank to databases. The global importance is computed using Global ObjectRank, which is the keyword-independent version of ObjectRank.

Motivating Example Consider the example of Figure 2.1, which illustrates a small subset of the DBLP database in the form of a labeled graph (author, conference and year nodes except for "R. Agrawal", "ICDE" and "ICDE 1997" respectively are omitted to simplify the figure). Schema graphs, such as the one of Figure 2.3, describe the structure of database graphs. Given a keyword query, e.g. the single-keyword query "OLAP", ObjectRank sorts the database objects by their relevance with respect to the user-provided keywords. Figure 2.2 illustrates the top-10 "OLAP" papers produced by our online demo available at two mirror sites, `http://www.db.ucsd.edu/ObjectRank` and `http://dbir.cis.fiu.edu/BibObjectRank/`. Notice that many entries (the "Data

Cube" and the "Modeling Multidimensional Databases" papers in Figure 2.1) of the top-10 list do not contain the keyword "OLAP" ("OLAP" is not even contained in their abstracts) but they clearly constitute important papers in the OLAP area, since they may be referenced by other papers of the OLAP area or may have been written by authors who have written other important "OLAP" papers.

Key Intuition of ObjectRank Conceptually, the ranking is produced in the following way: Myriads of random surfers are initially found at the objects containing the keyword "OLAP", which are called the base set, and then they traverse the database graph. In particular, at any time step a random surfer is found at a node and either (i) makes a move to an adjacent node by traversing an edge, or (ii) jumps randomly to an "OLAP" node without following any of the links. The probability that a particular traversal happens depends on multiple factors, including the type of the edge (in contrast to the Web link-based search systems [BP98, Hav02, RD02]). These factors are depicted in an authority transfer schema graph. Figure 2.4 illustrates the authority transfer schema graph that corresponds to the setting that produced the results of Figure 2.2. Assuming that the probability that the surfer moves back to an "OLAP" node is 15% (damping factor–random jump probability–[BP98]), the collective probability to move to a referenced paper is up to $85\% \times 70\%$ (70% is the authority transfer rate of the citation edge as explained below), the collective probability to move to an author of the paper is up to $85\% \times 20\%$, the probability to move from the paper to the forum where the paper appeared is up to $85\% \times 10\%$, and so on. As is the case with the PageRank algorithm as well, as time goes on, the expected percentage of surfers at each node v converges (Section 2.1) to a limit $r(v)$. Intuitively, this limit is the ObjectRank of the node.

An alternative way to conceive the intuition behind ObjectRank is to consider that authority/importance flows in the database graph in the same fashion that [Kle99] defined authority-based search in arbitrary graphs. Initially the "OLAP" authority is found at the objects that contain the keyword "OLAP". Then authority/importance flows, following the rules in the authority transfer schema graph, until an equilibrium is established that specifies that a paper is authorita-



1	**Implementing Data Cubes Efficiently.** *SIGMOD Conference* 1996. Venky Harinarayan, Anand Rajaraman, Jeffrey D. Ullman
2	**An Overview of Data Warehousing and OLAP Technology.** *SIGMOD Record* 1997. Surajit Chaudhuri, Umeshwar Dayal
3	**Index Selection for OLAP.** *ICDE* 1997. Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, Jeffrey D. Ullman
4	**On the Computation of Multidimensional Aggregates.** *VLDB* 1996. Sameet Agarwal, Rakesh Agrawal, Prasad Deshpande, Ashish Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, Sunita Sarawagi
5	**Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total.** *ICDE* 1996. Adam Bosworth, Jim Gray, Andrew Layman, Hamid Pirahesh
6	**Summarizability in OLAP and Statistical Data Bases.** *SSDBM* 1997. Hans-Joachim Lenz, Arie Shoshani
7	**Modeling Multidimensional Databases.** *ICDE* 1997. Rakesh Agrawal, Ashish Gupta, Sunita Sarawagi
8	**OLAP, Relational, and Multidimensional Database Systems.** *SIGMOD Record* 1996. George Colliat
9	**OLAP and Statistical Databases: Similarities and Differences.** *PODS* 1997. Arie Shoshani
10	**OLAP and Statistical Databases: Similarities and Differences.** *CIKM* 1996. Arie Shoshani

Figure 2.2: Top 10 papers on "OLAP" returned by ObjectRank

tive if it is referenced by authoritative papers, is written by authority authors and appears in authority conferences. Vice versa, authors and conferences obtain their authority from their papers. Notice that the amount of authority flow from, say, paper to cited paper or from paper to author or from author to paper, is arbitrarily set by a domain expert and reflects the semantics of the domain. For example, common sense says that in the bibliography domain a paper obtains very little authority (or even none) by referring to authoritative papers. On the contrary it obtains a lot of authority by being referred by authoritative papers.

Global ObjectRank is query-independent and is obtained by placing all nodes of the data graph in the base set.

Novel Properties of ObjectRank Keyword search in databases has some unique characteristics, which make the straightforward application of the random walk model as described in previous work [BP98, Hav02, RD02] inadequate. First, every database has different semantics, which ObjectRank uses to improve the quality of the keyword search. In particular, unlike the Web, where all edges are hyperlinks, the database schema exhibits the types of edges, and the attributes of the nodes. Note that previous works [RD02, CDG$^+$98] assign weights on the edges of the data graph according to the relevance of the incident nodes' text to the keywords. In contrast, ObjectRank assigns authority transfer rates on the schema graph, which captures the semantics of the database, since the relevance factor is reflected in the selection of the base set. Using the schema, ObjectRank specifies the ways in which authority flows across the nodes of the database graph. For example, the results of Figure 2.2 were obtained by annotating the schema graph of Figure 2.3 with the authority flow information that appears in Figure 2.4.

Furthermore, previous work [BP98, Hav02, RD02] assumes that, when calculating the global importance (in the ObjectRank framework, Balmin et al. make a clear distinction between the global importance of a node and its relevance to a keyword query), the random surfer has the same probability to start from any page p of the base set (this probability is called *base ObjectRank* of p). However, this is not true for every database. For example, consider a product complaints database. In this case, the business value of a customer can be represented by assigning to his/her node a base ObjectRank proportional to his/her total sales amount.

Another novel property of ObjectRank is adjustability, which allows for the tuning of the system according to the domain- and/or user-specific requirements. For example, for a bibliographic database, a new graduate student desires a search system that returns the best reading list around the specified keywords, whereas a senior researcher looks for papers closely related to the keywords, even if they are not of a high quality. These preference scenarios are made possible by adjusting the weight of the global importance versus the relevance to the keyword query. Changing the damping factor d offers another calibration opportunity. In particular, larger values of d favor nodes pointed by high-authority nodes, while smaller values of d favor nodes containing the actual keywords (that is, nodes in the base set). The handling of queries with multiple keywords offers more flexibility to the system as is described in Section 4.3. For example, a user may want to assign a higher weight to the relevance of a node to an infrequent keyword.

Performance of ObjectRank On the performance level, calculating the ObjectRank and Global ObjectRank values in runtime is a computationally intensive operation, especially given the fact that multiple users query the system. In [BHP04], this is resolved by precomputing inverted indexes where for each keyword the system materializes a sorted lists of the nodes with non-trivial scores for this keyword. During run-time ObjectRank employs the *Threshold Algorithm* [FLN01] to efficiently combine the lists. However, this approach induces

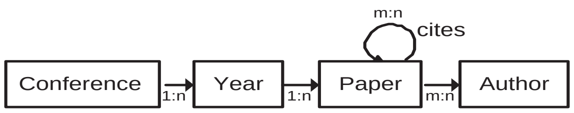


Figure 2.3: The DBLP schema graph.

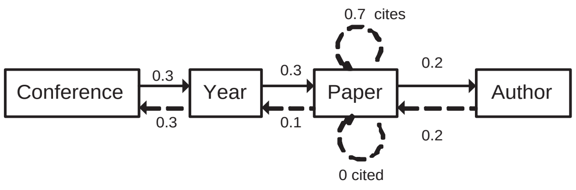


Figure 2.4: The DBLP authority transfer schema graph.

the cost of precomputing and storing the inverted index. Regarding the space requirements, notice that the number of keywords of a database is typically less than the number of users in a personalized search system [JW03]. Furthermore, ObjectRank does not store nodes with ObjectRank below a threshold value (chosen by the system administrator), which offers a space versus precision tradeoff.

Notice that the naive approach would be to calculate each keyword-specific ObjectRank (the same applies for Inverse ObjectRank) separately. Balmin et al. [BHP04] have found that it is substantially more efficient to first calculate the Global ObjectRank, and use these scores as initial values for the keyword-specific computations. This accelerates convergence, since in general, objects with high Global ObjectRank, also have high keyword-specific ObjectRank.

2.3 Data Model of ObjectRank

In this section, the essential definitions of ObjectRank [BHP04] are presented, that depict how ObjectRank captures the authority flow between objects in a database using a labeled graph. They are later used to define ranking metrics of ObjectRank [BHP04].

2.3.1 Modeling Database as Graph

ObjectRank [BHP04] views a database as a labeled graph, which is a model that easily captures both relational and XML databases. The *data graph* $D(V_D, E_D)$ is a labeled directed graph where every node v has a label $\lambda(v)$ and a set of keywords. For example, the node "ICDE 1997" of Figure 2.1 has label "Year" and the set of keywords $\{$`''ICDE'', ''1997'', ''Birmingham''`$\}$. Each node represents an *object* of the database and may have a sub-structure. Without loss of generality, ObjectRank assumes that each node has a tuple of attribute name/attribute value pairs. For example, the "Year" nodes of Figure 2.1 have `name`, `year` and `location` attributes. Notice that the keywords appearing in the attribute values comprise the set of keywords associated with the node. One may assume richer semantics by including the metadata of a node in the set of keywords. For example, the metadata "Forum", "Year", "Location" could be included in the keywords of a node. The specifics of modeling the data of a node are orthogonal to ObjectRank and will be neglected in the rest of the discussion.

Each edge e from u to v is labeled with its *role* $\lambda(e)$ (overload λ) and represents a relationship between u and v. For example, every "paper" to "paper" edge of Figure 2.1 has the label "cites". When the role is evident and uniquely defined from the labels of u and v, the edge label is omitted. For simplicity ObjectRank assumes that there are no parallel edges and an edge e from u to v is often denoted as "$u \to v$".

A critical issue in constructing the data graph for a database is to decide the granularity of the information in the nodes. For example, if we are to return a paper, should we also return the author names and the conference where the paper was published? ObjectRank adopts the idea of predefined "answer nodes" as described in [BNH$^+$02, DEGP98, GSBS03, HPB03][1]. Hence, in the above example, ObjectRank chooses to store the author and conference information in every paper node. Keep in mind that the data graph is a conceptual structure, so the actual physical storage may vary.

[1]In XKeyword [HPB03] they are referred to as target objects.

Data Graph	*Nodes*	*Edges*
Relational Database	Tuples (or attribute values)	Primary-to-Foreign Key Relationships
XML Database	XML Elements (or XML Nodes)	Containment or IR-IDREF Edges
Web	Pages	Hyperlinks

Figure 2.5: Mapping of Common Data Models to a Data Graph.

The data graph can represent relational [ACD02, HP02] and XML [HPB03, GSBS03] databases, as well as the Web [BP98]. The mappings of these data models to nodes and edges of the data graph are shown in Figure 2.5.

The use of ObjectRank ranking metrics does not require the existence of a schema. However, if a schema is present then it can be used to easier define the authority transfer rates (see below). Furthermore, the schema may offer optimization opportunities. The *schema graph* $G(V_G, E_G)$ (Figure 2.3) is a directed graph that describes the structure of D. Every node has an associated label. Each edge is labeled with a role, which may be omitted, as discussed above for data graph edge labels. It is said that a data graph $D(V_D, E_D)$ *conforms* to a schema graph $G(V_G, E_G)$ if there is a unique assignment μ of data-graph nodes to schema-graph nodes and a consistent assignment of edges such that:

1. for every node $v \in V_D$ there is a node $\mu(v) \in V_G$ such that $\lambda(v) = \lambda(\mu(v))$;
2. for every edge $e \in E_D$ from node u to node v there is an edge $\mu(e) \in E_G$ that goes from $\mu(u)$ to $\mu(v)$ and $\lambda(e) = \lambda(\mu(e))$.

Authority Transfer Schema Graph. From the schema graph $G(V_G, E_G)$, ObjectRank creates the *authority transfer schema graph* $G^A(V_G, E^A)$ to reflect the authority flow through the edges of the graph. This may be either a trial and error process, until the administrator is satisfied with the quality of the results, or a domain expert's task. In particular, for each edge $e_G = (u \to v)$ of E_G, two *authority transfer edges*, $e_G^f = (u \to v)$ and $e_G^b = (v \to u)$ are created. The two edges carry the label of the schema graph edge and, in addition, each one is annotated with a (potentially different) *authority transfer rate* - $\alpha(e_G^f)$ and $\alpha(e_G^b)$ correspondingly. In [BHP04], it is said that a data graph conforms to an authority transfer schema graph if it conforms to the corresponding schema graph. (Notice that the authority transfer schema graph has all the information of the original schema graph.)

Figure 2.4 shows the authority transfer schema graph that corresponds to the schema graph of Figure 2.3 (the edge labels are omitted). The motivation for defining two edges for each edge of the schema graph is that authority potentially flows in both directions and not only in the direction that appears in the schema. For example, a paper passes its authority to its authors and vice versa. Notice however, that the authority flow in each direction (defined by the authority transfer rate) may not be the same. For example, a paper that is cited by important papers is clearly important but citing important papers does not make a paper important.

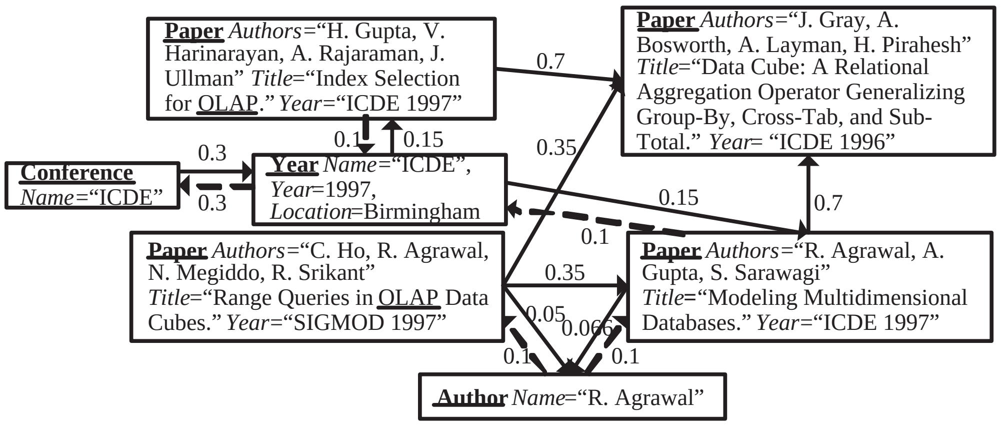
Notice that the sum of authority transfer rates of the outgoing edges of a schema node u may be less than 1[2], if the administrator believes that the edges starting from u do not transfer much authority. For example, in Figure 2.4, conferences only transfer 30% of their authority.

Authority Transfer Data Graph. Given a data graph $D(V_D, E_D)$ that conforms to an authority transfer schema graph $G^A(V_G, E^A)$, ObjectRank derives an *authority transfer data graph* $D^A(V_D, E_D^A)$ (Figure 2.6) as follows. For every edge $e = (u \rightarrow v) \in E_D$ the authority transfer data graph has two edges $e^f = (u \rightarrow v)$ and $e^b = (v \rightarrow u)$. The edges e^f and e^b are annotated with authority transfer rates $\alpha(e^f)$ and $\alpha(e^b)$. Assuming that e^f is of type e_G^f, then

$$\alpha(e^f) = \begin{cases} \frac{\alpha(e_G^f)}{OutDeg(u, e_G^f)}, \ if \ OutDeg(u, e_G^f) > 0 \\ 0, \ if \ OutDeg(u, e_G^f) = 0 \end{cases} \tag{2.4}$$

where $OutDeg(u, e_G^f)$ is the number of outgoing edges from u, of type e_G^f. The authority transfer rate $\alpha(e^b)$ is defined similarly. Figure 2.6 illustrates the authority transfer data graph that corresponds to the data graph of Figure 2.1 and the authority schema transfer graph of Figure 2.4. Notice that the sum of authority

[2]In terms of the random walk model, this would be equivalent to the disappearance of a surfer.



Paper Authors="H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman" Title="Index Selection for OLAP." Year="ICDE 1997"
Paper Authors="J. Gray, A. Bosworth, A. Layman, H. Pirahesh" Title="Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total." Year= "ICDE 1996"
Conference Name="ICDE"
Year Name="ICDE", Year=1997, Location=Birmingham
Paper Authors="C. Ho, R. Agrawal, N. Megiddo, R. Srikant" Title="Range Queries in OLAP Data Cubes." Year="SIGMOD 1997"
Paper Authors="R. Agrawal, A. Gupta, S. Sarawagi" Title="Modeling Multidimensional Databases." Year="ICDE 1997"
Author Name="R. Agrawal"
0.7
0.1
0.15
0.3
0.3
0.35
0.15
0.7
0.1
0.35
0.05
0.066
0.1
0.1

Figure 2.6: Authority transfer data graph

transfer rates of the outgoing edges of a node u of type $\mu(u)$ may be less than the sum of authority transfer rates of the outgoing edges of $\mu(u)$ in the authority transfer schema graph, if u does not have all types of outgoing edges.

2.3.2 Assigning Weights to Edge Types

Role of Edge Weights As described throughout this section, the semantic connections between objects are modeled as edges in a corresponding data graph in the ObjectRank framework [BHP04]. Since ObjectRank calculates authority flows between connected objects for ranking, it is very important to annotate edges with adequate weights to reflect the actual strength of connections. This enables ObjectRank to generate semantically meaningful search results.

In particular, ObjectRank aims to capture the query-independent strength of semantic connections. Notice that edges of different edge types may transfer different amount of authority. By assigning different edge weights to different edge types, ObjectRank can capture important domain knowledge such as "a paper cited by important papers is important, but citing important papers should not boost the importance of a paper". For example, in the DBLP dataset, suppose that a new paper p_{new} on "OLAP" cites $p_{classic}$, a classic paper in the "OLAP" research. Then, one can add two edges between p_{new} and $p_{classic}$, that are typed "Cites"($p_{new} \xrightarrow{cites} p_{classic}$) and "CitedBy"($p_{classic} \xrightarrow{citedBy} p_{new}$) respectively. Obviously, the amounts of authority flow via these two edges are not the same. If p_{new} becomes authoritative by getting many citations from authoritative papers, then $p_{classic}$ will become more authoritative. However, p_{new} should not get some of the authority of $p_{classic}$ simply by citing $p_{classic}$. This semantic knowledge can be captured by assigning 0 as the weight of "citedBy" edges, while annotating "cites" edges with nonzero weights.

Assigning Edge Weights Note that previous works [RD02, CDG+98] assign weights on the edges of the data graph according to the relevance of the incident nodes' text to the keywords, which is query-specific. In contrast, ObjectRank assigns authority transfer rates on the schema graph, which captures the semantics of the database, since the relevance factor is reflected in the selection of the base set of the query processing stage. Using the schema, ObjectRank specifies the ways in which authority flows across the nodes of the database graph. For example, the results of Figure 2.2 were obtained by annotating the schema graph of Figure 2.3 with the authority flow information that appears in Figure 2.4.

In [BHP04], ObjectRank assumes the process of assigning weights to each edge type as either a trial and error process until an administrator is satisfied with the quality of the results, or a domain expert's task. In this thesis, we also assume that it is beyond the scope of this thesis, and weights of each edge type are given.

In fact, there are several prior work that propose learning based approaches to automatically assign weights to each edge type, instead of using manually tuned weights. [NZWM05, CA06] proposed weight learners based on the partial rankings (pairwise preferences) of objects given by domain experts. They explore the search space of possible combinations of weights and iteratively reduce the difference between the rankings provided by domain experts and those generated by using the learned edge weights.

Another line of work such as Précis[SKI08] exploits query logs to compute weights of each edge type, considering query logs as a source for relevance feedback data from users.

2.4 Answering Queries Using ObjectRank

2.4.1 Answering Queries With Single Keyword

ObjectRank. In [BHP04], ObjectRank for a single keyword is defined as follows. In Section 4.3.3 this definition is extended to multiple keywords. Given a single keyword query w, ObjectRank finds the *keyword base set* $S(w)$ (from now on referred to simply as base set when the keyword is implied) of objects that contain the keyword w and assigns an ObjectRank $r^w(v_i)$ to every node $v_i \in V_D$ by resolving the equation

$$\mathbf{r}^w = d\mathbf{A}\mathbf{r}^w + \frac{(1-d)}{|S(w)|}\mathbf{s} \tag{2.5}$$

where $A_{ij} = \alpha(e)$ if there is an edge $e = (v_j \rightarrow v_i)$ in E_D^A and 0 otherwise, d controls the base set importance, and $\mathbf{s} = [s_1, \ldots, s_n]^T$ is the base set vector for $S(w)$, i.e., $s_i = 1$ if $v_i \in S(w)$ and $s_i = 0$ otherwise.

The damping factor d determines the portion of ObjectRank that an object transfers to its neighbors as opposed to making a random jump to one of the base set pages. It was first introduced in the original PageRank paper [BP98], where it was used to ensure convergence in the case of PageRank sinks. However, in addition to that, in ObjectRank it is used as a calibrating factor, since by decreasing d, ObjectRank favors objects that actually contain the keywords (i.e., are in base set) as opposed to objects that acquire ObjectRank through the incoming edges. The value for d used by PageRank [BP98] is 0.85, which ObjectRank also adopts to balance the importance of containing the actual keywords as opposed to being pointed by nodes containing the keywords.

Global ObjectRank. The definition of global ObjectRank is different for different applications or even users of the same application. In [BHP04], ObjectRank focuses on cases where the global ObjectRank is calculated applying the random surfer model, and including all nodes in the base set. The same calibrating parameters are available, as in the keyword-specific ObjectRank. Notice that this way of calculating the global ObjectRank, which is similar to the PageRank approach

[BP98], assumes that all nodes (pages in PageRank) initially have the same value. However, there are many applications where this is not true.

2.4.2 Answering Queries With Multiple Keywords

To reach a ranking function for node v given a multiple-keyword query "$q = \{w_1, \ldots, w_m\}$", Balmin et al. [BHP04] find the score $r^{w_i}(v)$ of v for every keyword w_i that is defined in Section 2.4.1, and then combine these scores (and possibly Global ObjectRank $r^G(v)$) to compute the final score $r^q(v)$.

In [BHP04], the semantics of a multiple-keywords query "$q = \{w_1, \ldots, w_m\}$" is defined by naturally extending the multiple-keywords random walk model. In particular, for the case of ObjectRank [BHP04] considers m independent random surfers, where the ith surfer starts from the keyword base set $S(w_i)$. For AND semantics, the ObjectRank of an object v with respect to the m-keywords query is the probability that, at a given point in time, the m random surfers are simultaneously at v.

$$r^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} r^{w_i}(v). \tag{2.6}$$

For OR semantics, the ObjectRank of v is the probability that, at a given point in time, *at least one* of the m random surfers will reach v.

$$r^{w_1,w_2}(v) = r^{w_1}(v) + r^{w_2}(v) - r^{w_1}(v)r^{w_2}(v) \tag{2.7}$$

and for more than two it is defined accordingly, as specified by the inclusion-exclusion principle (also known as the sieve principle). Notice that [Hav02] also takes the sum of the topic-sensitive PageRank values to calculate the PageRank of a page.

Weigh keywords by frequency. A drawback of the *combining function* of Equation 2.6 is that it favors the more popular keywords in the query. The reason is that the distribution of ObjectRank values is more skewed when the size $|S(w)|$ of the base set $S(w)$ increases, because the top objects tend to receive more references. For example, consider two results for the query "XML AND Index" shown

(a)

47.31	11.44	An XML Indexing Structure with Relative Region Coordinate. Dao Dinh Kha, ICDE 2001
41.02	3.08	DataGuides: Enabling Query ... Optimization in Semistructured... Roy Goldman, VLDB 1997
7.44	28.43	Access Path Selection in a RDBMS. Patricia G. Selinger, SIGMOD 1979
31.44	3.24	Querying Object-Oriented Databases. Michael Kifer, SIGMOD 1992
26.73	3.09	A Query … Optimization Techniques for Unstructured Data. Peter Buneman, SIGMOD 1996

(b)

47.31	11.44	An XML Indexing Structure with Relative Region Coordinate. Dao Dinh Kha, ICDE 2001
7.44	28.43	Access Path Selection in a RDBMS. Patricia G. Selinger, SIGMOD 1979
2.04	102.1	R-Trees: A Dynamic Index Structure for Spatial Searching. Antonin Guttman, SIGMOD 1984
1.73	112.7	The K-D-B-Tree: A Search Structure For Large … Indexes. John T. Robinson, SIGMOD 1981
41.02	3.08	DataGuides: Enabling Query … Optimization in Semistructured... Roy Goldman, VLDB 1997

Figure 2.7: Top 5 papers on "XML Index", with and without emphasis on "XML"

in Figure 2.7. Result (b) corresponds to the model described above. It noticeably favors the "Index" keyword over the "XML". The first paper is the only one in the database that contains both keywords in the title. However, the next three results are all classic works on indexing and do not apply directly to XML. Intuitively, "XML" as a more specific keyword is more important to the user. Indeed, the result of Figure 2.7 (a) was overwhelmingly preferred over the result of Figure 2.7 (b) by participants of the relevance feedback survey in [BHP04] (Section 2.5.1). The latter result contains important works on indexing in semistructured, unstructured, and object-oriented databases, which are more relevant to indexing of XML data. This result is obtained by using the modified formula:

$$r^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} (r^{w_i}(v))^{g(w_i)} \tag{2.8}$$

where $g(w_i)$ is a *normalizing exponent*, set to $g(w_i) = 1/log(|S(w_i)|)$. This exponent plays a role similar to the inverse document frequency (idf) in traditional Information Retrieval. Using the normalizing exponents $g(\text{"XML"})$ and $g(\text{"Index"})$ in the above example is equivalent to running in parallel $g(\text{"XML"})$ and $g(\text{"Index"})$ random walks for the "XML" and the "Index" keywords respectively. Balmin et al. [BHP04] propose the following combining function to incorporate the global ObjectRank of v

$$r^{q,G}(v) = (r^q(v)) \cdot (r^G(v))^g \tag{2.9}$$

where g is the $g$lobal ObjectRank weight, that determines the importance of the global ObjectRank.

2.5 Quality of ObjectRank

2.5.1 Qualitative Evaluation of ObjectRank

To evaluate the quality of the results of ObjectRank, Balmin et al. [BHP04] conducted two surveys. The first was performed on the DBLP database, with eight professors and Ph.D. students from the UC, San Diego database lab, who were not involved with the project. The second survey used the publications database of the IEEE Communications Society (COMSOC) [3] and involved five senior Ph.D. students from the Electrical Engineering Department.

Each participant was asked to compare and rank two to five lists of top-10 results for a set of keyword queries, assigning a score of 1 to 10, according to the relevance of the results list to the query. Each result list was generated by a different variation of the ObjectRank algorithm. One of the results lists in each set was generated by the "default" ObjectRank configuration which used the authority transfer schema graph of Figure 2.4 and $d = 0.85$. The users knew nothing about the algorithms that produced each result list. The survey was designed to investigate the quality of ObjectRank when compared to other approaches or when changing the adjusting parameters.

Effect of keyword-specific ranking. First, [BHP04] assesses the basic principle of ObjectRank, which is the keyword-specific scores. In particular, [BHP04] compared the default (that is, with the parameters set to the values discussed in Section 2.2) ObjectRank with the global ObjectRank ranking algorithm that sorts objects that contain the keywords according to their global ObjectRank (where the base-set contains all nodes). Notice that this is equivalent to what Google used to[4] do for Web pages, modulo some minor difference on the calculation of the relevance score by Google. The DBLP survey included results for two keyword queries: "OLAP" and "XML". The score was 7:1 and 5:3 in favor of the keyword-specific

[3]http://www.comsoc.org

[4]Google's current ranking algorithm is not disclosed.

ObjectRank for the first and second keyword query respectively. The COMSOC survey used the keywords "CDMA" and "UWB (ultra wideband)" and the scores were 4:1 and 5:0 in favor of the keyword-specific approach respectively.

Effect of authority transfer rates. [BHP04] compared results of the default ObjectRank with a simpler version of the algorithm that did not use different authority transfer rates for different edge types, i.e., all edge types were treated equally. In the DBLP survey, for both keyword queries, "OLAP" and "XML", the default ObjectRank won with scores 5:3 and 6.5:1.5 (the half point means that a user thought that both rankings were equally good) respectively. In the COMSOC survey, the scores for "CDMA" and "UWB" were 3.5:1.5 and 5:0 respectively.

Effect of the damping factor d. [BHP04] tested three different values of the damping factor d: 0.1, 0.85, and 0.99, for the keyword queries "XML" and "XML AND Index" on the DBLP dataset. Two points were given to the first choice of a user and one point to the second. The scores were 2.5 : 8 : 13.5 and 10.5 : 11.5 : 2 (the sum is 24 since there are 8 users times 3 points per query) respectively for the three d values. [BHP04] sees that higher d values are preferred for the "XML", because "XML" is a very large area. In contrast, small d are preferable for "XML AND Index", because few papers are closely related to both keywords, and these papers typically contain both of them. The results were also mixed in the COMSOC survey. In particular, the damping factors 0.1, 0.85, and 0.99 received scores of 5:6:4 and 4.5:3.5:7 for the queries "CDMA" and "UWB" respectively.

Effect of changing the weights of the keywords. [BHP04] compared the combining functions for AND semantics of Equations 2.6 with the weighted combining method described in Section 2.4.2 for the two-keyword queries "XML AND Index" and "XML AND Query", in the DBLP survey. The use of the normalizing exponents proposed in Section 2.4.2 was preferred over the simple product function with ratios of 6:2 and 6.5:1.5 respectively. In the COMSOC survey, the same experiment was repeated for the keyword query "diversity combining". The use of normalizing exponents was preferred at a ratio of 3.5:1.5.

2.5.2 Effectiveness of ObjectRank

To our best knowledge, there has been no previously reported experiments that directly compare the quality of top-K keyword search results by ObjectRank [BHP04] with annotated answers provided by domain experts. Nevertheless, we can still see the effectiveness of ObjectRank as below.

First, the relevance feedback survey results(Section 2.5.1) of [BHP04] show that results of the default ObjectRank are more semantically meaningful than those of 1) Google-like method that identifies objects using a simple IR function and then sorts by global ObjectRank scores or 2) ObjectRank over a data graph with equal-weight edges. [BHP04] could not analyze the quality of ObjectRank quantitatively because there was no available annotated graph-structured dataset for evaluating top-K keyword queries.

The quality of ObjectRank is re-evaluated in [HHP08]: we compare the original ObjectRank [BHP04] with various ranking schemes that combine ObjectRank and our new specificity measure, Inverse ObjectRank, in order to demonstrate the semantic contribution of this thesis. The results of the three qualitative experiments we performed are provided in Section 4.5.

Link-based ranking methods have been shown to produce effective search results in various research areas such as the personal information management domain [MCN06, Min07]. [MCN06, Min07] perform three email-related tasks - disambiguating person names, threading, and finding email aliases - using an authority-based approach very similar to ObjectRank over a data graph modeling an email corpus with learned edge weights. [MCN06, Min07] provide empirical results that show the superiority of the quality of authority-based ranking with learned edge weights over baseline approaches, a TFIDF method in [MCN06] and an authority-based approach with fixed edge weights in [Min07].

Since the effectiveness of link-based ranking methods including ObjectRank has already been demonstrated in various ways as discussed in this section, we do not revisit this issue in the following chapters. Note that the qualitative evaluation of ObjectRank [BHP04](and variations of personalized PageRank) is beyond the scope of this thesis.

Parts of Chapter 2 were published in Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD-2006), pp 796-798 and ACM Transactions on Database Systems 2008, 33(1) (TODS-2008). Heasoo Hwang, Vagelis Hristidis, and Yannis Papakonstantinou, “ObjectRank: a system for authority-based search on databases” and Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou, “Authority-based keyword search in databases”. The dissertation author and Vagelis Hristidis were the primary investigators and authors of these papers.

Chapter 3

BinRank: Fast Dynamic Authority-Based Search Using Materialized SubGraphs

3.1 Introduction

Recently, *dynamic* versions of the PageRank algorithm have become popular. They are characterized by a query-specific choice of the random walk starting points. In particular, two algorithms have gotten a lot of attention: personalized PageRank (PPR) for Web graph datasets ([Hav02, JW03, FRCS05, ALNO07]), and ObjectRank for graph-modeled databases ([BHP04, NZWM05, Cha07, HBPR07, HHP08]).

Personalized PageRank is a modification of PageRank that performs search personalized on a preference set that contains web pages that a user likes. For a given preference set, PPR performs a very expensive fixpoint iterative computation over the entire Web graph, while it generates personalized search results. Therefore, the performance issue of PPR has attracted a lot of attention ([JW03, FRCS05, ALNO07]). As mentioned earlier in Chapter 2, ObjectRank [BHP04] extends personalized PageRank to perform keyword search in databases.

Even though ObjectRank is an effective search method generating high recall search results(Section 2.5), it is not feasible to use ObjectRank to process keyword queries over a real-world dataset since ObjectRank suffers from the same performance issue as personalized PageRank as it requires multiple iterations over all nodes and links of the entire database graph. The original ObjectRank system has two modes, on-line and off-line. The on-line mode runs the ranking algorithm once the query is received, which takes too long on large graphs. For example, on a graph of articles of English Wikipedia[1] with 3.2 million nodes and 109 million links, even a fully optimized in-memory implementation of ObjectRank takes 20-50 seconds to run, as shown in Figure 3.2. In the off-line mode, ObjectRank precomputes top-k results for a query workload in advance. This precomputation is very expensive and requires a lot of storage space for precomputed results. Moreover, this approach is not feasible for all terms outside the query workload that a user may search for, i.e. for all terms in the dataset dictionary. For example, on the same Wikipedia dataset, the full dictionary precomputation would take about a CPU-year.

In this chapter, we introduce a BinRank system that employs a hybrid approach where query time can be traded off for preprocessing time and storage. BinRank closely approximates ObjectRank scores by running the same ObjectRank algorithm on a small subgraph, instead of the full data graph. The subgraphs are precomputed offline. The precomputation can be parallelized with linear scalability. For example, on the full Wikipedia dataset, BinRank can answer any query in less than 1 second, by precomputing about a thousand subgraphs, which takes only about 12 hours on a single CPU.

BinRank query execution easily scales to large clusters by distributing the subgraphs between the nodes of the cluster. This way, more subgraphs can be kept in RAM, thus decreasing the average query execution time. Since the distribution of the query terms in a dictionary is usually very uneven, the throughput of the system is greatly improved by keeping duplicates of popular subgraphs on multiple nodes of the cluster. The query term is routed to the least busy node that has the

[1]http://en.wikipedia.org

corresponding subgraph.

There are two dimensions to the subgraph precomputation problem: (1) how many subgraphs to precompute, and (2) how to construct each subgraph that is used for approximation. The intuition behind our approach is that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms. For (1), we group all terms into a small number (around 1,000 in case of Wikipedia) of "bins" of terms based on their co-occurrence in the entire dataset. For (2), we execute ObjectRank for each bin using the terms in the bins as random walk starting points and keep only those nodes that receive non-negligible scores.

Our experimental evaluation highlights the tuning of the system needed to balance the query performance with size and number of the precomputed subgraphs. Intuitively, query performance is highly correlated to the size of the subgraph, which in turn is highly correlated with the number of documents in the bin. Thus, normally, it is sufficient to create bins with a certain size limit to achieve a specific target running time. However there is some variability in the process and some bins may still result in unusually large subgraphs and slow queries. To address this, we employ an adaptive iterative process that further splits the problematic subgraphs to guarantee that a vast majority of queries will be executed within the allotted time budget.

Other approximation techniques have been considered before to improve the performance of dynamic authority-based search algorithms. Monte Carlo algorithms are introduced in [FRCS05] and [ALNO07] for approximation during precomputation. HubRank [Cha07] uses the same approximation as [FRCS05], but performs precomputation only for "hub" nodes. Other techniques might also suggest sampling-based techniques on-line. However, although these techniques claim on-line query processing, they have only been demonstrated on graphs with less than 10^6 links. In contrast, we demonstrate superior performance of our approach on a Wikipedia graph that is 2 orders of magnitude larger. We also show that our approximation using ObjectRank itself is more precise than the sampling-based techniques.

The contributions of this chapter are:

- The idea of approximating ObjectRank by using Materialized SubGraphs (MSG), which can be precomputed off-line to support on-line querying for a specific query workload, or the entire dictionary.
- Use of the default ObjectRank [BHP04] algorithm itself to generate MSGs for "bins" of terms.
- A greedy algorithm that minimizes the number of bins by clustering terms with similar posting lists.
- Extensive experimental evaluation on the Wikipedia dataset that supports our performance and search quality claims. The evaluation demonstrates superiority of BinRank over other state-of-the-art approximation algorithms.

The rest of the chapter is organized as follows: First, we start with the motivation of this chapter in Section 3.2 and give a survey of related work in Section 3.3. Then, the concept of materialized subgraphs is introduced in Section 3.4, and the bin construction algorithm is described in Section 3.5. In Section 3.6, we suggest the adaptive MSG re-computation method that improves the performance of BinRank. Section 3.7 describes the architecture of the BinRank system. Section 3.8 walks through the experimental evaluation. We conclude in Section 3.9.

3.2 Motivation

In this section, we discuss the result quality and performance issues that motivate this chapter.

ObjectRank returns top-k search results for a given query using both the content and the link structure in G. Since it utilizes the link structure that captures the semantic relationships between objects, an object that does not contain a given keyword but is highly relevant to the keyword can be included in the top-k list. This is in contrast to the static PageRank approach, that only returns objects containing the keyword sorted according to their PageRank score. This key difference is one of the main reasons for ObjectRank's superior result quality, as demonstrated by the relevance feedback survey reported in [BHP04].

However, the iterative computation of ObjectRank vectors described in Section 2.4 is too expensive to execute at run time. For a given query, ObjectRank iterates over the entire graph G to calculate the ObjectRank vector $\mathbf{r}$ until $|r_i^{(k+1)} - r_i^{(k)}|$ is less than the convergence threshold for every $r_i^{(k+1)}$ in $\mathbf{r^{(k+1)}}$ and $r_i^{(k)}$ in $\mathbf{r^{(k)}}$. This is a very strict stopping condition. This iterative computation may take a very long time if G has a large number of nodes and edges. Therefore, instead of evaluating a keyword query at query time, the original ObjectRank system [BHP04] precomputes the ObjectRank vectors of keywords in H, the set of keywords, during the preprocessing stage and then stores a list of <*ObjId, RankValue*> pairs per keyword. However, the preprocessing stage of ObjectRank is expensive, as it requires $|H|$ ObjectRank executions and $O(|V| \cdot |H|)$ bits of storage. In fact, according to the worst-case bounds for PPR index size proven in [FRCS05], the index size must be $\Omega(|V| \cdot |H|)$ bits, for any system that returns the exact ObjectRank vectors.

3.3 Related Work

The performance issue of Personalized PageRank (PPR) [JW03] has attracted a lot of attention. PPR performs a very expensive fixpoint iterative computation over the entire graph, while it generates personalized search results. To avoid the expensive iterative calculation at run time, one can naively precompute and materialize all the possible personalized PageRank vectors (PPVs). Although this method guarantees fast user response time, such precomputation is impractical as it requires a huge amount of time and storage especially when done on large graphs. In this section, we examine *hub-based* and *Monte Carlo* methods that address the performance problem of PPR, and we give an overview of HubRank [Cha07], that integrates the two approaches to improve the performance of ObjectRank. Even though these approaches enabled PPR to be executed on large graphs, they either limit the degree of personalization or deteriorate the quality of the top-k result lists significantly.

Hub-based Methods. *Hub-based approaches* materialize only a selected subset of PPVs. Topic-sensitive PageRank [Hav02] suggests materialization of 16 PPVs of selected topics and linearly combining them at query time. The personalized PageRank computation suggested in [JW03] enables a finer-grained personalization by efficiently materializing significantly more PPVs (e.g. 100K) and combining them using the hub decomposition theorem and dynamic programming techniques. However, it is still not a fully personalized PageRank, because it can personalize only on a preference set subsumed within a hub set H.

Monte Carlo Methods. *Monte Carlo methods* replace the expensive power-iteration algorithm with a randomized approximation algorithm ([FRCS05, ALNO07]). In order to personalize PageRank on any arbitrary preference set with maintaining just a small amount of precomputed results, [FRCS05] introduces the *fingerprint* algorithm that simulates the random walk model of PageRank and stored the ending nodes of sampled walks. Since each random walk is independent, fingerprint generation can be easily parallelized and the quality of search results improves as the number of fingerprints increases. However, as mentioned in [FRCS05], the precision of search results generated by the fingerprint algorithm is somewhat less than that of power-iteration-based algorithms, and sometimes the quality of its results may be inadequate especially for nodes that have many close neighbors. In [ALNO07], a Monte Carlo algorithm that takes into account not only the last visited nodes, but all visited nodes during the sampled walks, is proposed. Also, it showed that Monte Carlo algorithms with iterative start outperform those with random start.

HubRank. *HubRank* [Cha07] is a search system based on ObjectRank that improved the performance of ObjectRank by combining the above two approaches. It first selects a fixed number of hub nodes by using a greedy hub selection algorithm that utilizes a query workload in order to minimize the query execution time. Given a set of hub nodes H, it materializes the fingerprints of hub nodes in H. At query time, it generates an active subgraph by expanding the baseset with its neighbors. It stops following a path when it encounters a hub node whose PPV was materialized, or the distance from the baseset exceeds a fixed maximum length. HubRank recursively approximates PPVs of all active nodes, terminating with computation of PPV for the query node itself. During this computation the PPV approximations are dynamically pruned in order to keep them sparse. As stated in [Cha07], the dynamic pruning takes a key role in outperforming ObjectRank by a noticeable margin. However, by limiting the precision of hub vectors, HubRank may get somewhat inaccurate search results, as stated in [Cha07]. Also, since it materialized only PPVs of H, just as [JW03], the efficiency of query processing and the quality of query results are very sensitive to the size of H and the hub selection scheme. Finally, [Cha07] did not show any large-scale experimental results to verify the performance of HubRank.

Comparative Evaluation of BinRank. In our experiments section, we perform quality and performance experiments on the full English Wikipedia dataset exported in October 2007, to show that BinRank is an efficient ObjectRank approximation method that generates a high quality top-k list for any keyword query in the corpus.

For comparative evaluation of the performance of BinRank, we implemented the Monte Carlo algorithm 4 in [ALNO07]; that was shown to outperform other variations in [ALNO07]. We also implemented HubRank[Cha07] to check its performance on our Wikipedia dataset. To compare the quality(closeness of approximation) of BinRank results with those of previous work, we compute the similarity between top-K lists obtained by each method (i.e., approximate top-K lists) and top-K lists by ObjectRank over the given data graph (i.e., accurate top-K lists).

Unlike [FRCS05] which proves the convergence to the exact solution on arbitrary graphs, and [Cha07] and [JW03] which offer exact methods at the expense of limiting the choice of personalization, our solution is entirely heuristic. However, extensive experimental evaluation confirms that on real-world graphs BinRank can strike a good balance between query performance and closeness of approximation.

3.4 Relevant Subgraphs

Our goal is to improve the performance of ObjectRank while maintaining the high quality of top-k result lists. We focus on the fact that ObjectRank does not need to calculate the exact full ObjectRank vector **r** to answer a top-k keyword query ($K \ll |V|$). We identify three important properties of ObjectRank vectors that are directly relevant to the result quality and the performance of ObjectRank. First, for many of the keywords in the corpus, the number of objects with non-negligible ObjectRank values is much less than $|V|$. This means, that just a small portion of G is relevant to a specific keyword. Here, we say an ObjectRank value of v, $r(v)$, is non-negligible if $r(v)$ is above the convergence threshold. The intuition for applying the threshold is that differences between the scores that are within the threshold of each other are noise after ObjectRank execution. Thus, scores below threshold are effectively indistinguishable from zero, and objects that have such scores are not at all relevant to the query term. Second, we observed that top-k results of any keyword term t generated on subgraphs of G composed of nodes with non-negligible ObjectRank values, with respect to the same t, are very close to those generated on G. Third, when an object has a non-negligible ObjectRank value for a given baseset BS_1, it is guaranteed that the object gains a non-negligible ObjectRank score for another baseset BS_2 if $BS_1 \subseteq BS_2$. Thus, a subgraph of G composed of nodes with non-negligible ObjectRank values, with respect to a union of basesets of a set of terms, could potentially be used to answer any one of these terms.

Based on the above observations, we speed up the ObjectRank computation for query term q, by identifying a subgraph of the full data graph that contains all the nodes and edges that contribute to accurate ranking of the objects with respect to q. Ideally, every object that receives a non-zero score during the ObjectRank computation over the full graph should be present in the subgraph and should receive the same score. In reality, however, ObjectRank is a search system that is typically used to obtain only the top-k result list. Thus, the subgraph only needs to have enough information to produce the same top-k list. We shall call such a subgraph a Relevant subgraph (RSG) of a query.

Definition 3.4.1. *The top-k result list of the ObjectRank of keyword term t on data graph $G(V, E)$, denoted $OR(t, G, k)$, is a list of k objects from V sorted in descending order of their ObjectRank scores with respect to a baseset that is the set of all objects in V that contain keyword term t.*

Definition 3.4.2. *A Relevant Sub-Graph ($RSG(t, G, k)$) of a data graph $G(V, E)$ with respect to a term t and a list size k is a graph $G_s(V_s, E_s)$, such that $V_s \subset V$, $E_s \subset E$, and $OR(t, G, k) = OR(t, G_s, k)$.*

It is hard to find an exact RSG for a given term, and it is not feasible to precompute one for every term in a large workload. However, we introduce a method to closely approximate RSGs. Furthermore, we observed that a single subgraph can serve as an approximate RSG for a number of terms, and that it is quite feasible to construct a relatively small number of such subgraphs that collectively *cover*, i.e. serve as approximate RSGs, all the terms that occur in the dataset.

Definition 3.4.3. *An Approximate Relevant Sub-Graph ($ARSG(t, G, k, c)$) of a data graph $G(V, E)$ with respect to a term t, list size k, and confidence limit $c \in [0, 1]$, is a graph $G_s(V_s, E_s)$, such that $V_s \subset V$, $E_s \subset E$, and $\tau(OR(t, G, k), OR(t, G_s, k)) > c$.*

Kendall's τ is a measure of similarity between two lists of [Ken55]. This measure is commonly used to describe the quality of approximation of top-k lists of exact ranking (R_E) and approximate ranking (R_A) that may contain ties (nodes with equal ranks) [FRCS05, Cha07]. A pair of nodes that is strictly ordered in both list is called *concordant* if both rankings agree on the ordering, and *discordant* otherwise. A pair is *e-tie*, if R_E does not order the nodes of the pair, and *a-tie*, if R_A does not order them. Let C, D, E, and A denote the number of concordant, discordant, e-tie, and a-tie pairs respectively. Then, Kendall's τ similarity between two rankings, R_E and R_A, is defined as $\tau(R_E, R_A) = \frac{C-D}{\sqrt{(M-E)(M-A)}}$, where M is the total number of possible pairs, $M = \frac{n(n-1)}{2}$ and $n = |R_E \cup R_A|$. We linearly scale τ to $[0, 1]$ interval as in [FRCS05, Cha07].

Definition 3.4.4. *An* ARSG cover *of a data graph $G(V, E)$, with respect to a keyword term workload W, list size k, and confidence limit $c \in [0, 1]$ is a set of graphs Γ, such that for every term $t \in W$, there exists $G_s \in \Gamma$ that is $ARSG(t, G, k, c)$, and inversely every $G_s \in \Gamma$ is an $ARSG(t, G, k, c)$ for at least one term $t \in W$.*

We construct an ARSG for term t by executing ObjectRank with some set of objects B as the baseset and restricting the graph to include only nodes with non-negligible ObjectRank scores $NOR(B)$, i.e. those above the convergence threshold ϵ_t of the ObjectRank algorithm. We call the induced subgraph $G[NOR(B)]$ a materialized subgraph for set B, denoted $MSG(B)$.

The main challenge of this approach is identifying a baseset B, that will provide a good RSG approximation for term t. We focus on sets B, that are supersets of the baseset of t. This relationship gives us the following important result.

Theorem 3.4.5. *If $BS_1 \subset BS_2$, then $(v \in MSG(BS_1) \Rightarrow v \in MSG(BS_2))$.*

Proof. Let BS_1 and BS_2 be subsets of V that satisfy $BS_1 \subset BS_2$. Also, let $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_{2\backslash 1}}$ be the ObjectRank vectors and $\mathbf{q_1}$, $\mathbf{q_2}$, and $\mathbf{q_{2\backslash 1}}$ be the normalized baseset vectors corresponding to BS_1, BS_2, and $(BS_2 - BS_1)$ respectively. Then, by applying the linearity theorem in [JW03] on the ObjectRank formula in Equation (2.5), we get the following equation[2]:
$\alpha_1\mathbf{r_1} + \alpha_{2\backslash 1}\mathbf{r_{2\backslash 1}} = d\mathbf{A}(\alpha_1\mathbf{r_1} + \alpha_{2\backslash 1}\mathbf{r_{2\backslash 1}}) + (1-d)(\alpha_1\mathbf{q_1} + \alpha_{2\backslash 1}\mathbf{q_{2\backslash 1}})$, where $\alpha_1 = \frac{|BS_1|}{|BS_2|}$ and $\alpha_{2\backslash 1} = \frac{|BS_2 - BS_1|}{|BS_2|}$. Since $BS_1 \subset BS_2$, $\alpha_1 + \alpha_{2\backslash 1} = 1$, which satisfies the linearity theorem. Notice that, since $\alpha_1\mathbf{q_1} + \alpha_{2\backslash 1}\mathbf{q_{2\backslash 1}} = \mathbf{q_2}$, $\alpha_1\mathbf{r_1} + \alpha_{2\backslash 1}\mathbf{r_{2\backslash 1}} = \mathbf{r_2}$ holds.
Now, let us consider a node $v \in G$ is in $MSG(BS_1)$. Since we just showed $\mathbf{r_2} = \alpha_1\mathbf{r_1} + \alpha_{2\backslash 1}\mathbf{r_{2\backslash 1}}$, $\mathbf{r_2}(v) = \alpha_1\mathbf{r_1}(v) + \alpha_{2\backslash 1}\mathbf{r_{2\backslash 1}}(v)$ also holds. Thus, $\mathbf{r_2}(v) \geq \alpha_1\mathbf{r_1}(v)$, because $\alpha_{2\backslash 1} > 0$ and $\mathbf{r_{2\backslash 1}}(v) \geq 0$. Also, since $v \in MSG(BS_1)$, $\mathbf{r_1}(v) > \frac{\epsilon}{|BS_1|}$ by definition of MSG.
Since $\alpha_1 = \frac{|BS_1|}{|BS_2|}$ and $\mathbf{r_1}(v) > \frac{\epsilon}{|BS_1|}$, $\mathbf{r_2}(v) \geq \alpha_1\mathbf{r_1}(v) > \frac{|BS_1|}{|BS_2|} \cdot \frac{\epsilon}{|BS_1|} = \frac{\epsilon}{|BS_2|}$. Since $\mathbf{r_2}(v) > \frac{\epsilon}{|BS_2|}$, by definition of MSG, $v \in MSG(BS_2)$. □

[2]For better presentation of this chapter, let $BS(w)$ and $\mathbf{q}$ denote $S(w)$ and $\frac{\mathbf{s}}{|S(w)|}$ of Equation (2.5) respectively

According to this theorem, for a given term t, if the term baseset $BS(t)$ is a subset of B, all the important nodes relevant to t are always subsumed within $MSG(B)$, i.e., all the non-negligible end points of random walks originated from starting nodes containing t are present in the subgraph generated using B.

However, notice that even though two nodes v_1 and v_2 are guaranteed to be found both in G and in $MSG(B)$, the ordering or their ObjectRank scores might not be preserved on $MSG(B)$ as we do not include intermediate nodes if their ObjectRank scores are below the convergence threshold. Missing intermediate nodes could deteriorate the quality of ObjectRank scores computed on $MSG(B)$. However, it is unlikely that many walks terminating on relevant nodes will pass through irrelevant nodes. Thus, even if $MSG(B, G)$ is not an $RSG(t, G, k)$, it is very likely to be $ARSG(t, G, k, c)$ with high confidence c. Our experimental evaluation supports this intuition.

In this chapter, we construct MSGs by clustering all the terms of the dictionary, or of a query workload if one is available, into a set of term "bins". We create a baseset B for every bin by taking the union of the posting lists of the terms in the bin, and construct $MSG(B)$ for every bin. We remember the mapping of terms to bins, and at query time, we can uniquely identify the corresponding bin for each term, and execute the term on the MSG of this bin.

Theorem 3.4.5 supports our intuition that a bin's MSG is very likely to be an ARSG for each term in the bin with fairly high confidence. Thus, the set of all bin MSGs will be an ARSG cover with sufficiently high confidence. Our empirical results support this claim. For example, after a reasonable tuning of parameter settings ($\epsilon = 0.0005$ and maximum B size of 4000 documents), 90% of our random workload terms ran on their respective bin MSGs with $\tau(OR(t, G, 100), OR(t, MSG, 100)) > 0.9$. Moreover, the other 10% of terms, which had $\tau_{100} < 0.9$, were all very infrequent terms. The most frequent among them appeared in 8 documents. τ_{100} tends to be relatively small for infrequent terms, because there simply may not be 100 objects with meaningful relationships to the baseset objects.

3.5 Bin Construction

As outlined above, we construct a set of MSGs for terms of a dictionary or a workload by partitioning the terms into a set of *term bins* based on their co-occurrence. We generate an MSG for every bin based on the intuition that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms.

There are two main goals in constructing term bins. First, controlling the size of each bin to ensure that the resulting subgraph is small enough for ObjectRank to execute in a reasonable amount of time. Second, minimizing the number of bins to save the preprocessing time. After all, we know that precomputing ObjectRank for all terms in our corpus is not feasible.

To achieve the first goal, we introduce a *maxBinSize* parameter that limits the size of the union of the posting lists of the terms in the bin, called *bin size*. As discussed above, ObjectRank uses the convergence threshold that is inversely proportional to the size of the baseset, i.e., the bin size in case of subgraph construction. Thus, there is a strong correlation between the bin size and the size of the materialized subgraph. As we show in Section 3.8, the value of *maxBinSize* should be determined by quality and performance requirements of the system.

The problem of minimizing the number of bins is NP-hard. In fact, if all posting lists are disjoint, this problem reduces to a classical NP-hard bin packing problem [GJ85]. We apply a greedy algorithm that picks an unassigned term with the largest posting list to start a bin and loops to add the term with the largest overlap with documents already in the bin. We use a number of heuristics to minimize the required number of set intersections, which dominate the complexity of the algorithm. The tight upper bound on the number of set intersections that our algorithm needs to perform is the number of pairs of terms that co-occur in at least one document. To speed-up the execution of set intersections for larger posting lists, we use KMV synopses [BHR+07] to estimate the size of set intersections.

The bin computation algorithm in Algorithm 1 works on term posting lists from a text index. As the algorithm fills up a bin, it maintains a list of document

Algorithm 1: PackTermsIntoBins

```
Input: A set of workload terms $W$, with their posting lists
Output: A set of bins $B$
while $W$ is not empty do
    create a new empty bin $b$;
    create an empty cache of candidate terms $C$;
    pick term $t \in W$ with the largest posting list size $|t|$;
    while $t$ is not null do
        add $t$ to $b$, and remove it from $W$;
        jd compute a set of terms $T$ that co-occur with $t$;
        foreach $t' \in T$ do
            insert (or update) mapping $< t', null >$ into $C$;
        $bestI := 0$;
        foreach mapping $< c, i > \in C$ do
            if $i = null$ then
                $i := |b \cap c|$;
                update mapping $< c, i >$ in $C$;
            $union := |b| + |c| - i$;
            if $union > maxBinSize$ then
                remove $< c, i >$ from $C$;
            else if $i > bestI$ then
                $bestI := i$;
                $t := c$;
        if $bestI = 0$ then
            pick $t \in W$ with maximum $|t| \leq maxBinSize - |b|$;
            if no such $t$ exists, $t := null$;
    add completed $b$ to $B$;
```

IDs, that are already in the bin, and a list of *candidate terms*, that are known to overlap with the bin (i.e. their posting lists contain at least one document, that was already placed into the bin). The main idea of this greedy algorithm is to pick a candidate term with a posting list that overlaps the most with documents already in the bin, without posting list union size exceeding the maximum bin size.

While it is more efficient to prepare bins for a particular workload that may come from a system query log, it is dangerous to assume that a query term that has not been seen before, will not be seen in the future. We demonstrate that it is feasible to use the entire dataset dictionary as the workload, in order to be able to answer any query.

Due to caching of candidate intersection results in lines 12-14 of the algorithm, the upper bound on the number of set intersections performed by this algorithm is the number of pairs of co-occurring terms in the dataset. Indeed, in the worst case, for every term t that has just been placed into the bin, we need to intersect the bin with every term t' that co-occurs with t, in order to check if t' is subsumed by the bin completely, and can be placed into the bin "for free".

For example, consider N terms with posting lists of size X each, that all co-occur in one document d_0 with no other co-occurrences. If maximum bin size is $2(X-1)$, a bin will have to be created for every term. However, to get to that situation, our algorithm will have to check intersections for every pair of terms. Thus, the upper bound on the number of intersections is tight.

In fact, it is easy to see from the above example that no algorithm that packs the bins based on the maximum overlap can do so with fewer than $N(N-1)/2$ set intersections in the worst case. Fortunately, real-world text databases have structures that are far from the worst case, as we show in Section 3.8.

Lastly, we show that the number of bins the algorithm uses to pack a set of posting lists is at most $2\alpha OPT$, where α indicates the degree of overlap across posting lists and OPT is minimal. Notice that, since BinRank constructs an MSG for each bin during preprocessing, $2\alpha OPT$ is also the upper bound of the number of MSGs.

Theorem 3.5.1. *Given a set of posting lists $\mathcal{S}$ of S_i's, suppose that there exists*

$\alpha \geq 1$ such that $\sum_{S_i \in \mathcal{S}} |S_i| \leq \alpha |\bigcup_{S_i \in \mathcal{S}} S_i|$. Then, the approximation ratio of PackTermsIntoBins is 2α.

Proof. Let OPT and OPT' denote the optimal number of bins and the number of bins *PackTermsIntoBins* uses.

- Claim1: $OPT \geq \frac{\sum_{S_i \in \mathcal{S}} |\bigcup S_i|}{maxBinSize \cdot \alpha}$

  Since no bin can hold a total capacity of more than *maxBinSize*, $OPT \geq \frac{|\bigcup_{S_i \in \mathcal{S}} S_i|}{maxBinSize}$. Also, since α satisfies $|\bigcup_{S_i \in \mathcal{S}} S_i| \geq \frac{\sum_{S_i \in \mathcal{S}} |S_i|}{\alpha}$, $OPT \geq \frac{|\bigcup_{S_i \in \mathcal{S}} S_i|}{maxBinSize} \geq \frac{\sum_{S_i \in \mathcal{S}} |\bigcup S_i|}{maxBinSize \cdot \alpha}$. $\therefore$ Claim1 holds.

- Claim2: $|\bigcup_{S_i \in \mathcal{S}} S_i| > (OPT' - 1) * \frac{maxBinSize}{2}$

  Since no more than one bin is less than half full, $|\bigcup_{S_i \in \mathcal{S}} S_i| > (OPT' - 1) * \frac{maxBinSize}{2}$. Also, since $\sum_{S_i \in \mathcal{S}} |S_i| \leq \alpha |\bigcup_{S_i \in \mathcal{S}} S_i|$ for $\alpha \geq 1$, $\sum_{S_i \in \mathcal{S}} |S_i| \leq |\bigcup_{S_i \in \mathcal{S}} S_i|$. $\therefore$ Claim2 holds.

By Claim1 and Claim2, $OPT \geq \frac{\sum_{S_i \in \mathcal{S}} |\bigcup S_i|}{maxBinSize \cdot \alpha} > \frac{OPT' - 1}{2\alpha}$, i.e., $OPT > \frac{OPT' - 1}{2\alpha}$. $\therefore$ $OPT' \leq 2\alpha OPT$ $\square$

3.6 Adaptive MSG Re-computation

We construct bins of up to a certain number of documents based on the intuition that a limited bin size will limit the resulting MSG size, which, in turn, will limit the running time of the query. As we demonstrate in Section 3.8 this intuition holds for the average case, however for a small minority of MSGs and queries the running time can be 2-3 times higher than the average. Fortunately, we can detect problematic MSGs and replace them with more efficient ones during the pre-processing stage.

Recall that the ObjectRank running time scales linearly with two parameters: the number of iterations required and the size of the graph. The number of iterations is correlated to the size of the baseset, so for a given MSG, queries with the largest basesets are going to be the slowest. And for queries with fixed sized basesets, the running time will largely depend on the number of links in the graph.

In fact, we report in Section 3.8.6 a 94% correlation between the number of links on an MSG and the BinRank running time for queries with large basesets. This observation enables us to reliably identify problematic MSGs based only on their link counts.

However, the correlation between the bin sizes and the MSG link counts is less obvious. Figure 3.14 shows that the link-count for MSGs follows a normal distribution even with all the Bin and MSG generation parameters fixed. Thus, setting the generation parameters in a way that no MSG exceeds a certain link-count threshold is not going to be practical. Instead, we set the parameters in such a way that only a small minority of MSGs exceeds the limit, and then deal with this minority separately.

One way to deal with dangerously large MSGs is to recompute them with a larger convergence threshold, thus making them smaller. However, this may diminish the subsequent query result quality, so instead we choose to keep the same ϵ, but regenerate the bins that produced these MSGs with a smaller *maxBinSize*.

To do this, we introduce a new threshold *maxMSGSize* and generate a set of rejected bins RB, that resulted in MSGs with the number of links larger than *maxMSGSize*. We then generate a new set of workload terms W', which consists of all the keywords of all bins in RB, and rerun the *PackTermsIntoBins* algorithm with W' and the new *maxBinSize* set to the half of the original one. The new set of bins replaces RB, and the new MSGs are produced and tested against the *maxMSGSize*. If some MSGs still fail the test, the process can be repeated iteratively.

3.7 System Architecture

Figure 3.1 shows the architecture of the BinRank system. During the preprocessing stage (left side of figure), we generate MSGs as defined in Section 3.4. During query processing stage (right side of figure), we execute the ObjectRank algorithm on the subgraphs instead of the full graph and produce high quality approximations of top-k lists at a small fraction of the cost. In order to save pre-

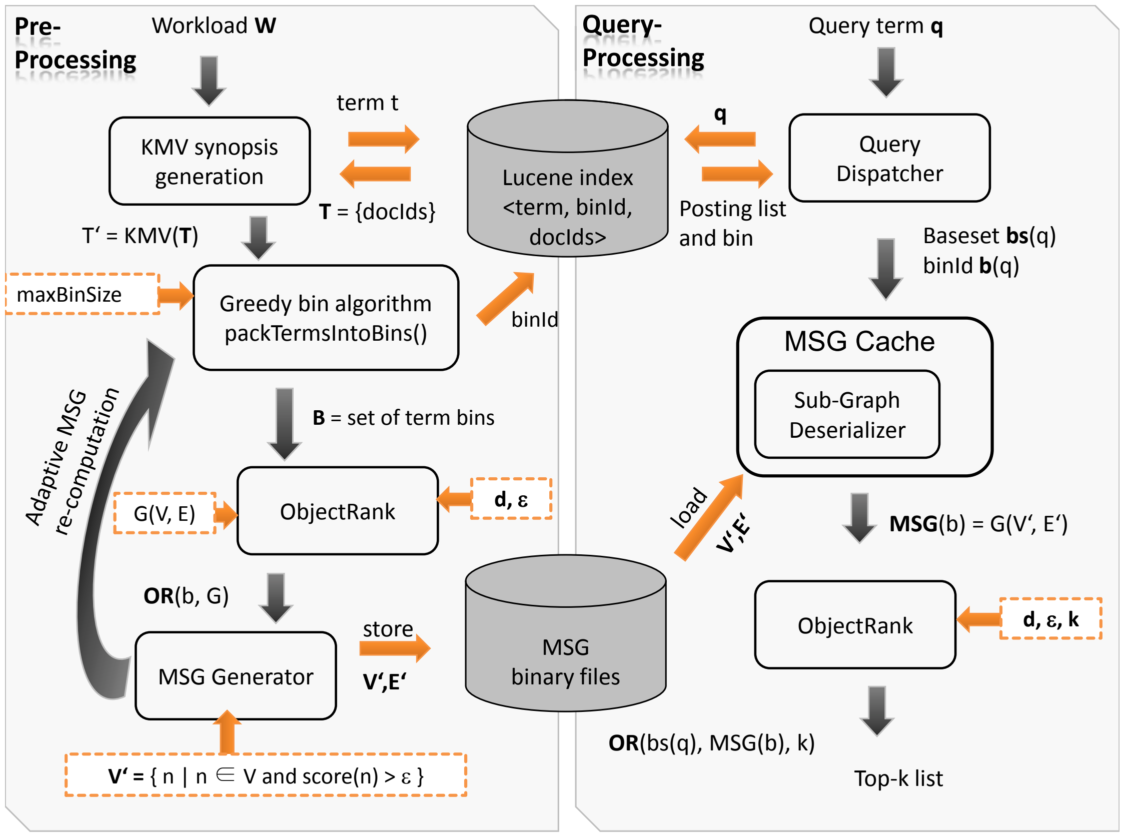


Figure 3.1: System Architecture

processing cost and storage, each MSG is designed to answer multiple term queries. We observed in the Wikipedia dataset that a single MSG can be used for 330 to 2000 terms, on average.

3.7.1 Preprocessing

The preprocessing stage of BinRank starts with a set of workload terms W for which MSGs will be materialized. If an actual query workload is not available, W includes the entire set of terms found in the corpus. We exclude from W all terms with posting lists longer than a system parameter *maxPostingList*. The posting lists of these terms are deemed too large to be packed into bins. We execute ObjectRank for each such term individually, and store the resulting top-k lists. Naturally, *maxPostingList* should be tuned so that there are relatively few of these frequent terms. In the case of Wikipedia, we used *maxPostingList* $= 2000$ and only 381 terms out of about 700000 had to be precomputed individually. This process took 4.6 hours on a single CPU.

For each term $w \in W$, BinRank reads a posting list T from the Lucene[3] index and creates a KMV synopsis T' that is used to estimate set intersections.

The bin construction algorithm, *PackTermsIntoBins*, partitions W into a set of bins composed of frequently co-occurring terms. The algorithm takes a single parameter *maxBinSize*, which limits the size of a bin posting list, i.e. the union of posting lists of all terms in the bin. During the bin construction, BinRank stores the bin identifier of each term into the Lucene index as an additional field. This allows us to map each term to the corresponding bin and MSG at query time.

The ObjectRank module takes as input a set of bin posting lists B and the entire graph $G(V, E)$ with a set of ObjectRank parameters, the damping factor d and the threshold value ϵ. The threshold determines the convergence of the algorithm as well as the minimum ObjectRank score of MSG nodes.

Our ObjectRank implementation stores a graph as a row-compressed adjacency matrix. In this format, the entire Wikipedia graph consumes 880MB of storage, and can be loaded into main memory for MSG generation. In case that the entire data graph does not fit in main memory, we can apply parallel PageRank computation techniques such as hypergraph partitioning schemes described in [BdJKT05].

The MSG generator takes the graph G and the ObjectRank result with respect to a term bin b, and then constructs a subgraph $G_b(V', E')$ by including only nodes with $r^t(u) \geq \epsilon_b$. ϵ_b is the convergence threshold of b, that is $\frac{\epsilon}{|BS(b)|}$. Given the set of MSG nodes V', the corresponding set of edges E' is copied from the in-memory copy of G. The edge construction takes 1.5 - 2 seconds for a typical MSG with about 5 million edges.

Once the MSG is constructed in memory, it is serialized to a binary file on disk in the same row-compressed adjacency matrix format to facilitate fast deserialization. We observed that de-serializing a 40MB MSG on a single SATA

[3]http://lucene.apache.org

disk drive takes about 0.6 seconds. In general, deserialization speed can be greatly improved by increasing the transfer rate of the disk subsystem.

3.7.2 Query Processing

For a given keyword query q, the query dispatcher retrieves from the Lucene index the posting list $bs(q)$ (used as the baseset for the ObjectRank execution) and the bin identifier $b(q)$. Given a bin identifier, the MSG mapper determines whether the corresponding MSG is already in memory. If it is not, the MSG deserializer reads the MSG representation from disk. The BinRank query processing module uses all available memory as an LRU cache of MSGs.

For smaller data graphs, it is possible to dramatically reduce MSG storage requirements by storing only a set of MSG nodes V', and generating the corresponding set of edges E' only at query time. However, in our Wikipedia dataset that would introduce an additional delay of 1.5 - 2 seconds, which is not acceptable in a keyword search system.

The ObjectRank module gets the in-memory instance of MSG, the baseset, and a set of ObjectRank calibrating parameters: (i) the damping factor d, (ii) the convergence threshold ϵ, and (iii) the number of top-k list entries k. Once the ObjectRank scores are computed and sorted, the resulting document ids are used to retrieve and present the top-k objects to the user.

Multi-keyword queries are processed as follows: For a given conjunctive query composed of n terms, $\{t_1, \ldots, t_n\}$, the ObjectRank module gets MSGs, $\{MSG(b(t_1)), \ldots, MSG(b(t_n))\}$, and evaluates each term over the corresponding MSG. Then, it multiplies the ObjectRank scores obtained over MSGs to generate the top-k list for the query. For a disjunctive query, the ObjectRank module sums the ObjectRank scores with respect to each term calculated using MSGs to produce BinRank scores.

One of the advantages of BinRank query execution engine is that it can easily utilize large clusters of nodes. In this case, we distribute MSGs between the nodes and employ Hadoop[4] to start an MSG cache and an ObjectRank engine

[4]http://hadoop.apache.org

Web service on every node. A set of dispatcher processes, each with its own replica of the Lucene index, route the queries to the appropriate nodes.

3.8 Experiments

We present our experimental evaluation in this section. We first describe our experimental setup using English Wikipedia articles. Then, we show performance numbers for ObjectRank followed by numbers for BinRank. Finally, we present a performance comparison of BinRank with Monte Carlo Method and HubRank.

3.8.1 Setup

We evaluate the performance of the BinRank algorithm on the collection of English Wikipedia articles exported in October 2007. We parsed the 13.8GB dump file and extracted 3.2M articles and 109M intra-wiki links of 10 types (e.g., "Regular links", "Category links", "See also links", etc.). All the experiments in this section are performed over the labeled graph $G_{wiki} = (V_{wiki}, E_{wiki})$, that is composed of the Wikipedia articles as nodes and the intra-wiki links as edges. We used the standard row-compressed matrix format to represent the link structure and weight dissipation rates of E_{wiki} compactly. We were able to store the $3.2M * 3.2M$ transition matrix of G_{wiki} with 109M non-zero elements in only 880MB. We created a Lucene text index of the Wikipedia article titles, which takes up 154MB. The dictionary of the index contains 698,214 terms.

We chose to index only article titles, by analogy with the original ObjectRank [BHP04] setup that used only publication titles from DBLP. It is important for ObjectRank to have a baseset of objects that are highly related to a search term. However, a large article can mention a term without being meaningfully related to it. For that reason, title index works better than an index on the full text of the articles. In order to use the full article text index, the ObjectRank algorithm would have to be augmented to take into account Lucene search scores of the baseset documents. This is one of our future research directions.

For our experiments, we implemented the BinRank system (and other algorithms for performance comparisons) in Java and performed experiments on a single PC with a Pentium4 3.40GHz CPU and 2.0GB of RAM.

3.8.2 ObjectRank on the Full Wikipedia Graph

ObjectRank on G_{wiki} takes too long to be executed online, and it consumes around 880MB of memory just for the link information of G_{wiki}. As shown in Figure 3.2, it takes around 20-50 seconds (30 seconds on average) to compute the dynamically generated top-k list for a given single keyword query even with our optimized, in-memory ObjectRank execution engine. For frequent keywords, that have posting lists with over 200 documents, the ObjectRank is likely to take longer. Since frequent keywords are found in many articles, they are more likely to be meaningfully connected to many other articles through many paths, resulting in a wider search space for ObjectRank to evaluate and rank.

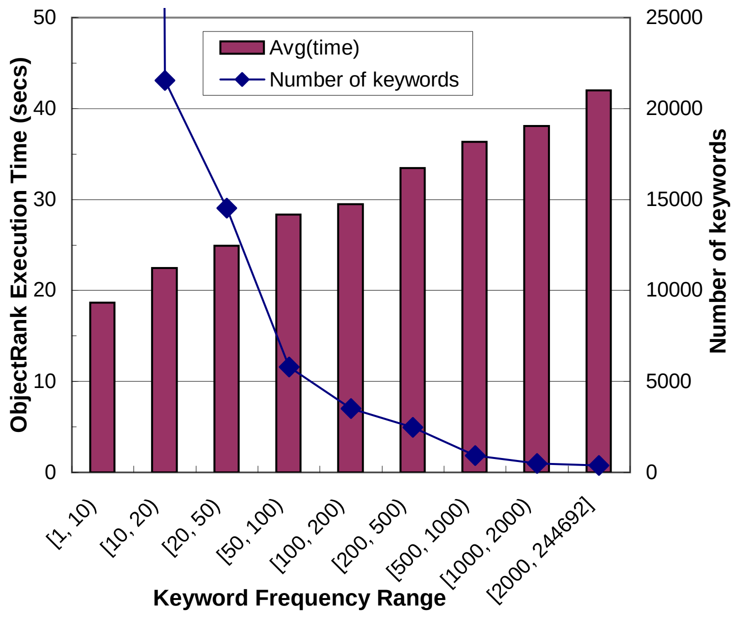


Figure 3.2: The number of keywords and average ObjectRank execution time on the Wikipedia graph per frequency range (ϵ is fixed to 5.0E-4)

Figure 3.2 also shows the keyword frequency distribution obtained from the Lucene text index built on the article titles. The total number of keywords in the index is 698214, and the keyword frequencies follows the typical power law distribution.

3.8.3 BinRank - Preprocessing Stage

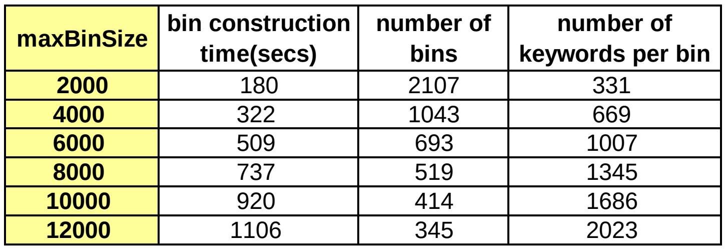
During the BinRank preprocessing stage, we generate bins for all the keywords in the corpus. Once the bins are constructed, we generate an MSG per bin by executing ObjectRank on G_{wiki} using the union of the posting lists of the terms in a bin as a single baseset. We first describe the performance of the bin construction and MSG generation in Section 3.8.3, and then measure the query result quality and the impact of two important parameters, ϵ and *maxBinSize*, in Section 3.8.4.

Bin Construction. To measure the performance of the bin construction stage, we examine the bin construction time and the number of bins constructed with different *maxBinSize* values.

We construct bins for all terms in our Lucene index, except for the 381 most frequent terms which have posting lists longer than a system parameter *maxPostingList* = 2000. Recall from Section 3.7, that such terms are deemed to be too frequent, so we precompute their ObjectRank authority vectors individually. This process takes 4.6 hrs.

To pack the remaining 697833 keywords into terms, we construct bins with various *maxBinSize*s as shown in Figure 3.3. Notice that as *maxBinSize* increases, the bin construction algorithm generates fewer bins while consuming more time. The running time goes up because the greedy algorithm needs to try more intersections of larger sets to fill the larger bins. However, even with *maxBinSize* = 12000, BinRank generates all 345 bins in only 1106 seconds. This is a small fraction of the total preprocessing time, which is dominated by MSG construction, as we will see next.

Note, that Wikipedia page titles are a very simple case for bin generation as the typical document size is extremely small. We also tested the bin construction algorithm on the full text of Wikipedia pages. In this case, the total size of the

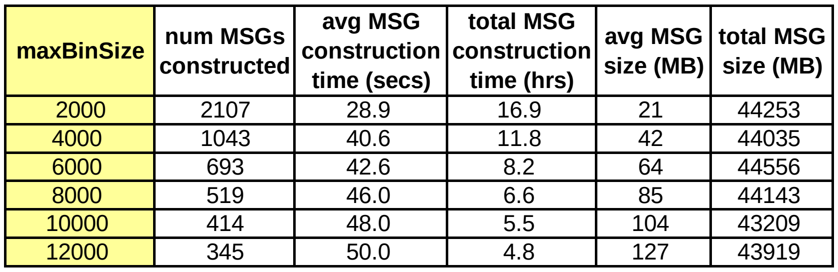


maxBinSize	bin construction time(secs)	number of bins	number of keywords per bin
2000	180	2107	331
4000	322	1043	669
6000	509	693	1007
8000	737	519	1345
10000	920	414	1686
12000	1106	345	2023

Figure 3.3: Performance of bin construction

posting lists in the text index was 84 million, vs. 4.8 million for titles. The algorithm produced 6340 bins with *maxBinSize* 5000, performing over 4 billion intersections. The packing process took about 70 hours.

MSG Generation. Once the bins are constructed, we generate an MSG for each bin. For our Wikipedia dataset, we generated a comprehensive set of MSGs with 24 combinations of the two parameters, *maxBinSize* and ϵ. For each combination, we measure the performance of BinRank, i.e. the query time and the quality of top-k lists.



maxBinSize	num MSGs constructed	avg MSG construction time (secs)	total MSG construction time (hrs)	avg MSG size (MB)	total MSG size (MB)
2000	2107	28.9	16.9	21	44253
4000	1043	40.6	11.8	42	44035
6000	693	42.6	8.2	64	44556
8000	519	46.0	6.6	85	44143
10000	414	48.0	5.5	104	43209
12000	345	50.0	4.8	127	43919

Figure 3.4: The effect of *maxBinSize* on the MSG construction cost (ϵ is fixed to 5.0E-4)

maxBinSize determines the number of bins to be constructed, and thus the number of MSGs generated (the 2nd column in Figure 3.4). The construction time and average size go up with the *maxBinSize*. Intuitively, the larger the baseset the more objects will be related to it. And the more objects have non-trivial scores, the more iterations it will take the ObjectRank algorithm to reach the fixpoint.

Figure 3.4 supports this intuition.

Notice that the total MSG construction time decreases significantly, as the *maxBinSize* increases. However, the average MSG size increases at the same time, which leads to slower query execution time. Thus, there is a clear tradeoff between preprocessing time and query time in BinRank.

Figure 3.5 shows the effect of ϵ on MSG construction time and the size of MSGs. Smaller ϵ implies that ObjectRank will need more iterations to reach the convergence point, and that more nodes will have scores above the bin convergence threshold $\epsilon_b = \frac{\epsilon}{BinSize}$. Thus, both construction time and MSG size decrease as the ϵ increases.

epsilon	num MSGs constructed	avg MSG construction time (secs)	total MSG construction time (hrs)	avg MSG size (MB)	total MSG size (MB)
2.5E-04	1043	46.5	13.5	73	75773
5.0E-04	1043	40.6	11.8	42	44035
1.0E-03	1043	38.7	11.2	25	25691
5.0E-03	1043	31.0	9.0	7	7324
1.0E-02	1043	29.0	8.4	4	4562

Figure 3.5: The effect of ϵ on the MSG construction cost (*maxBinSize* is fixed to 4000)

An interesting observation from Figures 3.4 and 3.5, is that the storage requirements of BinRank, i.e. the total size of MSGs, is controlled by the choice of ϵ and is virtually unaffected by *maxBinSize*. Of course, the quality of the BinRank's score approximations are also strongly affected by ϵ, as we show next. Thus, one has to strike a balance between the quality of results and the storage overhead. For example, BinRank produces extremely high quality results with ϵ =5.0E-4. However, this setup requires 44GB of storage for MSGs, which is 50 times of the size of G_{wiki}. Another way to approach this tradeoff is to say, that the amount of disk, or even better, RAM available to the system will determine the quality of results.

As we discussed in Section 3.7, it is possible to reduce MSGs storage requirements by materializing MSG nodes only and extract links at query time. The edge extraction adds 1.5-2 seconds to the query time, but the storage requirements in this case go down from 44GB to only 203MB, which is similar to the size of our Lucene index, 154MB.

3.8.4 BinRank - Query Processing Stage

Quality Measures. For a given keyword query, BinRank generates an approximate top-k list using the corresponding MSG. The exact top-k list is obtained by executing ObjectRank on G_{wiki} with small ϵ =1.0E-4. The two lists are compared using the same three quality measures as in [FRCS05]: RAG(relative aggregated goodness), *precision* at K, and *Kendall's* τ.

Let $OR(kw, K)$ and $BR(kw, K)$ denote the accurate top-k list by ObjectRank and the approximate top-k list by BinRank for a given keyword kw. In our experiments, both top-k lists are lists of Wikipedia article IDs sorted by the authority score. Let $ORScore(n, kw)$ denote the exact keyword-specific authority score of a node n computed by ObjectRank.

RAG and precision measure the quality of $BR(kw, K)$ by considering top-k lists as sets, say $ORSet(kw, K)$ and $BRSet(kw, K)$. RAG is the ratio of the aggregated exact authority scores of nodes in $BR(kw, K)$ to scores of nodes in $OR(kw, K)$. Precision at K computes the ratio of the size of intersection to K.

$$RAG(K) = \frac{\sum_{n \in BRSet(kw,K)} ORScore(n, kw)}{\sum_{n \in ORSet(kw,K)} ORScore(n, kw)}$$

$$Prec(K) = \frac{|BRSet(kw, K) \cap ORSet(kw, K)|}{K}$$

Kendall's τ, as defined in Section 3.4, compares the orderings of the top-k lists, i.e., $OR(kw, K)$ and $BR(kw, K)$. It is the most stringent quality measure of the three measures that we use. τ value of 1 means that the lists are identical, and 0 that they are disjoint or in inverse order.

Since we primarily aim to get high quality top-k lists within reasonable amount of query time, we want to find good combinations of *maxBinSize* and ϵ for BinRank. To tune these parameters, we compute quality measures for all 24 sets of MSGs described above, 6 different *maxBinSize* values and 4 different ϵ values.

The smallest *maxBinSize*, 2000, is chosen to be the same as the maximum posting list size for terms that are put into bins.

We run a workload of 92 randomly selected query terms on all of these 24 sets of MSGs.

Effect of maxBinSize on query time and quality of top-k lists using BinRank. With ϵ =5.0E-4, we generated MSGs with 6 different *maxBinSize* values starting from the smallest *maxBinSize*, 2000. Figure 3.6 shows, that query time increases linearly as *maxBinSize* increases. This is because the average size of MSGs also increases linearly as depicted in Figure 3.4. For example, when *maxBinSize* is 2000, an MSG is 21MB, but it increases to 42MB if *maxBinSize* increases to 4000.

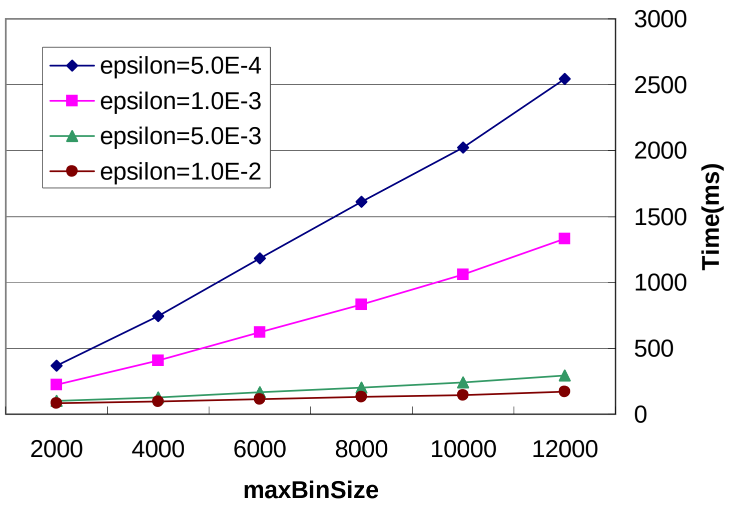


Figure 3.6: The effect of *maxBinSize* on the BinRank running time

Next, we investigate the effects of the MSG size, which is determined by the *maxBinSize*, on the accuracy of top-k lists. Figure 3.7 shows the average accuracy of top-100 lists measured by the three goodness measures given ϵ =5.0E-4. First, all the measures are in $[0.95, 1]$ range, indicating that the quality of the top-100 lists obtained by BinRank is very good. Second, as *maxBinSize* increases from 2000 to 12000, the accuracy remains the same or improves very slightly. However,

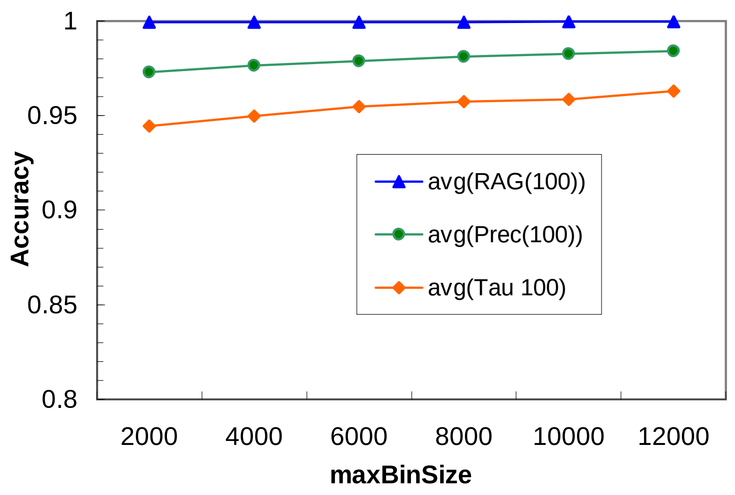


Figure 3.7: The effect of *maxBinSize* on the top-100 accuracy(ϵ is fixed to 5.0E-4)

we do not see a noticeable improvement on the quality of top-k lists. In contrast, the accuracy of top-k is sensitive to ϵ as is shown in Figure 3.10.

Figure 3.8 illustrates the relationship between *maxBinSize*, ϵ and the accuracy of top-k lists. It shows the distribution of τ_5 through τ_{1000} with 12 combinations of the parameters: all 6 different *maxBinSize* and 2 ϵ values, 5.0E-3 and 5.0E-4. One can see that the 12 line form two clusters, one for ϵ =5.0E-3(bottom) and the other for ϵ =5.0E-4(top).

For a given ϵ and a set of *maxBinSize* values, if larger *maxBinSize* does not improve quality of top-k lists in a big margin, then we do not see any good reason to increase *maxBinSize*. Actually, it decreases the preprocessing time in Figure 3.4 by reducing the number of MSGs, but increases the query processing time as is shown in Figure 3.6. For example, with ϵ =5.0E-4, we can see from Figure 3.4 that the average size of MSGs is 127MB when $maxBinSize = 12000$, while it is 42MB for $maxBinSize = 4000$. However, Figure 3.8 shows that the top-k lists generated on these two MSGs are very similar on average. We computed standard deviations of τ values of top-k lists with varying *maxBinSize* values and a fixed ϵ. They are very low: $stdev(\tau_{20}) = 0.00627$ and $stdev(\tau_{100}) = 0.00672$.

However, we cannot reduce *maxBinSize* without considering the total MSG construction time. One might want to construct bins with very small *maxBinSize*.

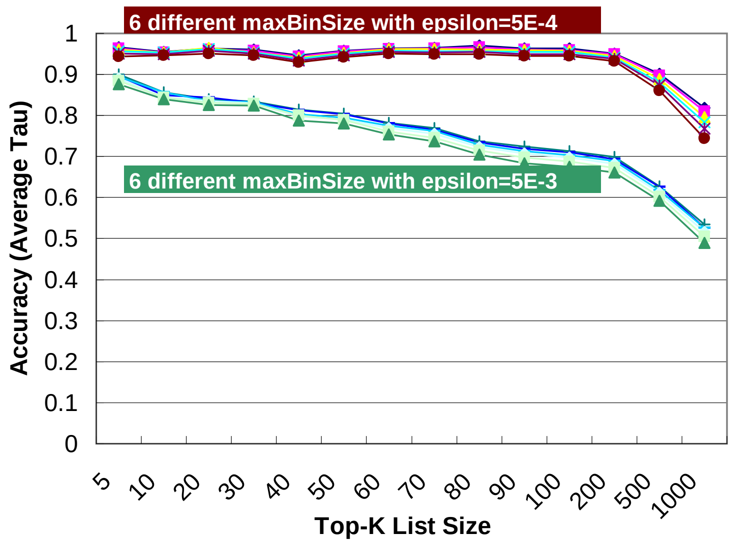


Figure 3.8: The effect of *maxBinSize* on the top-k accuracy with fixed ϵ

Setting aside the accuracy issue, BinRank will construct too many bins to complete MSGs construction stage in a given time budget. The extreme case is to precompute and materialize MSGs or authority vectors for all the keywords in the dictionary, which is infeasible especially when the size of the dictionary and the size of the full graph are huge as in our Wikipedia dataset.

Effect of ϵ on query time and quality of top-k lists of BinRank. As we observed in Figure 3.5, as ϵ increases, the average size of MSGs also increases. It takes more time to generate top-k lists on a larger MSG on average as shown in Figure 3.9.

Now, we analyze the effect of ϵ on the quality of top-k lists. Unlike *maxBinSize*, the quality of top-k lists improves as we can see in Figure 3.10.

To measure how much an MSG covers the context of keywords in the corresponding bin, we computed the *RankMass coverage metric*[CS07] of sets of MSGs generated with 5 different ϵ values. In our experiments, we define the RankMass of an MSG with respect to a keyword as the ratio of aggregated authority scores of nodes in the MSG to the sum of all authority scores in $G_{wiki} = (V_{wiki}, E_{wiki})$. Let $MSG(b, e, m)$ denote a set of nodes in the MSG generated for a bin b with $\epsilon = e$

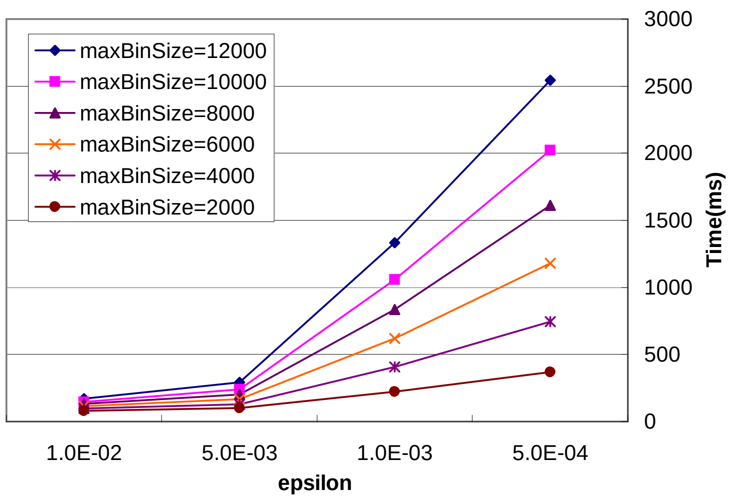


Figure 3.9: The effect of ϵ on the BinRank running time

and $maxBinSize = m$. Let us assume, that the bin b contains a workload keyword, kw. Then the RankMass of an MSG w.r.t. kw is: $RankMass(MSG(b, e, m), kw)$

$$= \frac{\sum_{v \in MSG(b,e,m)} OR(v, kw)}{\sum_{v \in V_{wiki}} OR(v, kw)}$$

We computed the average RankMass coverage of an MSG using all the keywords in our workload, which shows how well an MSG covers the context of keywords in the corresponding bin. As we can expect with an increasing ϵ, the RankMass also increases rapidly.

For example, if we compare two sets of MSGs constructed with $maxBinSize = 4000$ and $maxBinSize = 6000$,
$avg(|MSG(b, 5\text{E-}4, 12000)|) = 3 * avg(|MSG(b, 5\text{E-}4, 4000)|)$, but the average RankMass only increases by 5.7%. The average size of MSGs of $maxBinSize = 4000$ is 1.52% of $|V_{wiki}|$ while that of $maxBinSize = 6000$ is 4.59% of $|V_{wiki}|$. However, if we decrease ϵ from 1E-3 to 5E-4, the average size of MSGs increases from 0.98% of $|V_{wiki}|$ to 1.52% of $|V_{wiki}|$, while the RankMass increases by 7.0%.

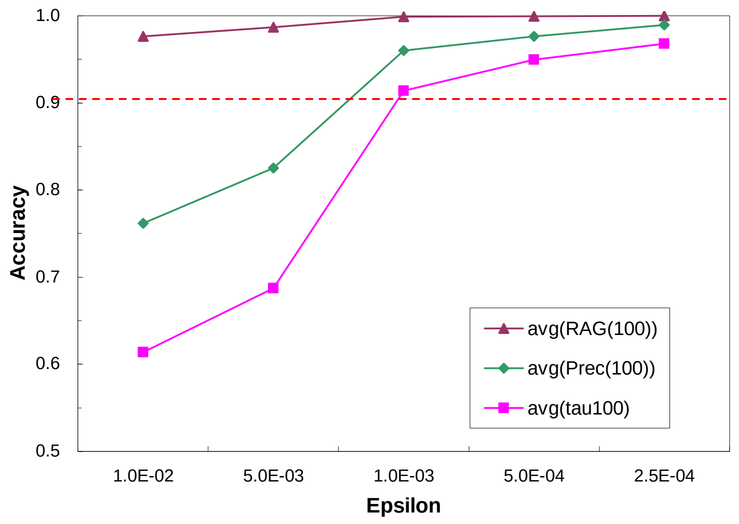


Figure 3.10: The effect of ϵ on the top-100 accuracy($maxBinSize$ is fixed to 4000)

3.8.5 BinRank for Multi-keyword Queries

In this section, we investigate the performance of BinRank for multi-keyword queries. Given a multi-keyword query q composed of n keywords, $k_1 \ldots k_n$, BinRank first evaluates each k_i over the MSG corresponding to the keyword, $MSG(k_i)$. Then, it combines the rank scores computed over those MSGs according to query semantics to produce the top-k list for q.

We observed from our experimental results that if a multi-keyword query contains highly relevant keywords such as "martial" AND "arts" or "fine" AND "performing", BinRank assigns those relevant keywords into the same bin, and thus evaluates those keywords using the same MSG. In this case, the top-k accuracy of the query is higher than randomly generated multi-keyword queries. However, if keywords composing a multi-keyword query are assigned to different bins and the query is conjunctive, BinRank has to evaluate each keyword over different MSGs and combine scores.We assign zero scores to nodes not in the MSG. Hence, if a conjunctive query contains keywords whose MSGs do not overlap, BinRank will return an empty result. However, we observed no such cases thorughout our experiments, because certain highly popular subgraphs of G_{wiki}, obtain non-negligible scores regardless of the keywords assigned to a bin.

We randomly generated 600 multi-keyword queries to measure the top-k accuracy of conjunctive queries and disjunctive queries containing 2 to 4 keywords. Throughout our experiments, we use $maxBinSize$ = 4000 and ϵ =5.0E-4. We do not report the statistics of the BinRank running time for multi-keyword queries, because it is dominated by the running time of BinRank for each individual query term.

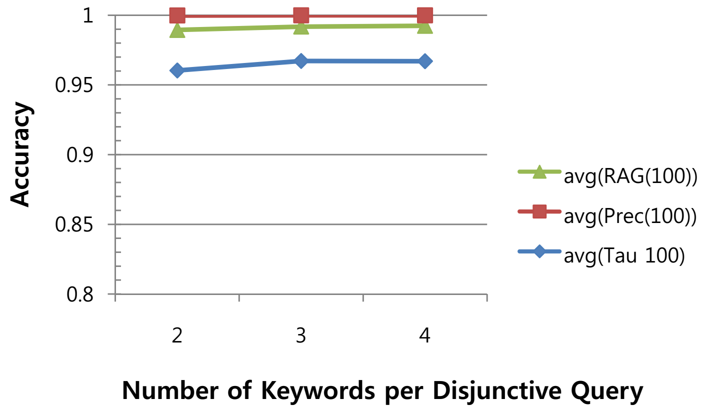


Figure 3.11: The top-100 accuracy for disjunctive queries ($maxBinSize$ = 4000 and ϵ =5.0E-4)

We can see from Figure 3.11 that the top-100 accuracy of disjunctive queries is higher than the top-100 accuracy for single-keyword queries shown in Figure 3.7 and Figure 3.10. As shown in Figure 3.8, the accuracy of a top-k list drops as K increases, because scores of highly ranked nodes are more stable than those of the rest. Since the top-100 list for a disjunctive query tends to include top-k nodes ($K \leq 100$) in the top-100 lists obtained over MSGs, its accuracy is at least as high as that for single-keyword queries or slightly higher than that.

As is shown in Figure 3.12, $RAG(100)$ and $Prec(100)$ are above 0.9, indicating that the top-100 lists obtained by BinRank include most of the nodes in the top-100 lists generated by ObjectRank over G_{wiki}. However, the average τ_{100} for conjunctive queries remains in $[0.75, 0.8]$ range, which is lower than those for single-keyword queries(Figure 3.7) or disjunctive queries(Figure 3.11). Therefore, we can see that for a given conjunctive query, BinRank generates a top-k list that contains all the nodes in the top-k list obtained over G_{wiki}, but the ordering of

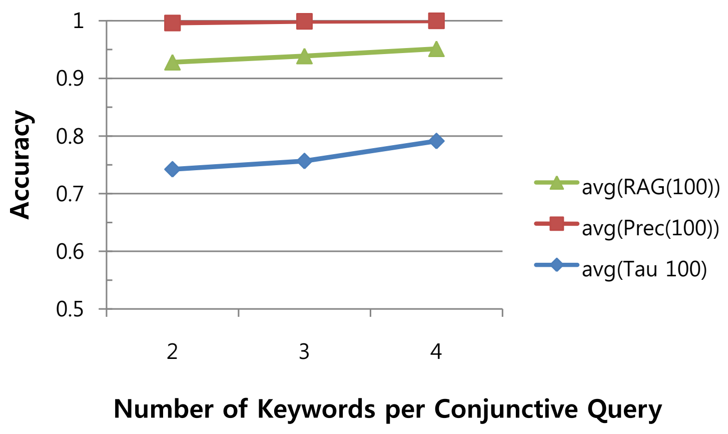


Figure 3.12: The top-100 accuracy for conjunctive queries ($maxBinSize$ = 4000 and ϵ =5.0E-4)

nodes in the BinRank top-k list is not highly accurate. This is mainly because the MSGs are not large enough to cover all the important paths through which significant amount of authority flows into or between the top-100 nodes, even though most of the links between top-100 nodes exist on the corresponding MSGs. To improve the top-k accuracy of conjunctive queries, we can increase the coverage of MSGs by using smaller ϵ. Note that increasing $maxBinSize$ does not improve the top-k accuracy in a big margin as shown in Figure 3.7.

However, with smaller ϵ, BinRank generates larger MSGs, increasing query execution time. Especially, we observed that some MSGs require unacceptably long running time. Given time budget, we want to identify such MSGs and re-compute them as described in Section 3.8.6.

3.8.6 Adaptive MSG Re-computation

In this section, we first want to examine the entire set of MSGs to understand the features of MSGs. We obtain 1043 bins and then generate a set of MSGs, $\mathcal{M}$, using BinRank parameters, $maxBinSize$ = 4000 and ϵ =5.0E-4. The average number of nodes and links on an MSG is 48616 and 5.2M, which is just 1.52% of $|V_{wiki}|$ and 4.83% of $|E_{wiki}|$. Recall that G_{wiki} has 3.2M nodes and 109M links.

Next, to evaluate the quality of the MSGs in $\mathcal{M}$, we pick a set of keywords,

$\mathcal{Q}$, by selecting the keyword with largest frequency among the keywords assigned to each bin. The range of keyword frequency of $\mathcal{Q}$ is $[1, 2000]$. We select the most frequent keyword for each bin since they are very likely to result in the slowest BinRank execution time out of all keywords in the bin, as discussed in Section 3.6.

The average BinRank execution time for queries in $\mathcal{Q}$ is 856ms, which is much faster than the average ObjectRank execution time on G_{wiki}, 30 seconds. However, we observe that some queries in $\mathcal{Q}$ require almost 2 seconds to evaluate, which is sometimes not acceptable. The goal of the MSG re-computation algorithm is to predict and prevent such cases, so that the BinRank running time does not exceed a certain time budget, which is set to 1 second throughout these experiments.

As we discussed in Section 3.6 the BinRank query running time depends on the features of the query (e.g. the baseset size) and those of the corresponding MSG (e.g. the number of nodes and the number of links). Other factors such as the connectivity of links on an MSG and the topology of baseset nodes also affect the BinRank running time, but they are harder to quantify, and the simple features prove to be sufficiently good predictors.

The correlation coefficients, denoted by r, between the BinRank running time and each of the three simple features are the followings:

- $r_1 = 0.564$: with the number of nodes on an MSG
- $r_2 = 0.700$: with the number of links on an MSG
- $r_3 = 0.459$: with the baseset size of a query

r_2 is noticeably higher than r_1 or r_3, which indicates that the number of links on an MSG is more tightly correlated to the BinRank running time than the other two features. Actually, since r_2 is obtained from all the queries in $\mathcal{Q}$ and their baseset sizes vary significantly within $[1, 2000]$, we can see the effect of the number of links on an MSG more clearly after reducing the effect of the baseset size. To do it, we select a set of 292 queries with high frequency, $[1000, 2000]$, and denote it as $\mathcal{Q}_{hf}$. From Figure 3.13 obtained using $\mathcal{Q}_{hf}$, we can clearly observe a very strong correlation between the number of links on an MSG and the BinRank running time. With very high R^2 value, the BinRank running time of a query is almost linear in the number of links on the corresponding MSG. Also, the correlation coefficient between the number of links on an MSG and the BinRank running time using high frequency keywords in $\mathcal{Q}_{hf}$ is 0.938, which also indicates a very strong correlation.

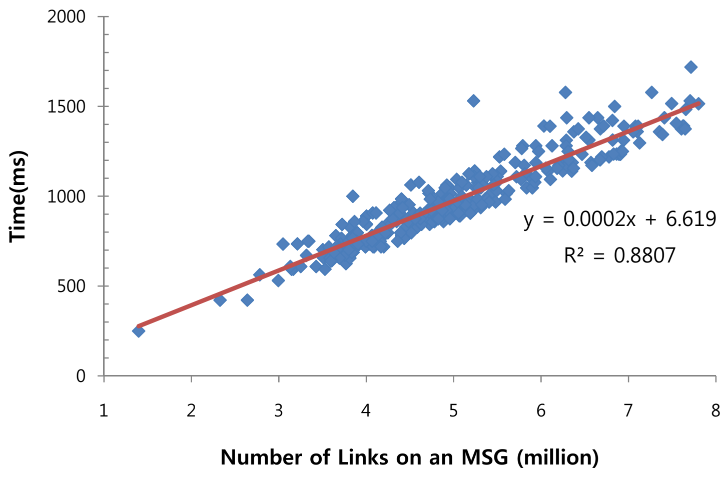


Figure 3.13: The effect of the number of links on an MSG on the BinRank running time (*maxBinSize* = 4000 and ϵ =5.0E-4). The Pearson correlation coefficient is 0.938.

By exploiting this strong correlation, we select MSGs whose BinRank running time will be above a certain time budget with high probability. As we can see in Figure 3.14, the number of links on an MSG almost follows a normal distribution $N(\mu, \sigma^2)$ where $\mu = 5.2E6$ and $\sigma = 1.0E6$. Our experiments show that among the 1043 MSGs in $\mathcal{M}$, 144 MSGs have more than $(\mu + \sigma)$ links, and among them, 138 MSGs(94.4%) require more than 1sec to produce top-k lists for the largest frequency keyword in the corresponding bin. In contrast, the probability that the worst case BinRank running time exceeds 1sec is just 16.4% for MSGs with less than $(\mu + \sigma)$ links. If we pick the MSGs with less than μ links, only 5.6% of them spend more than 1sec to compute top-k lists in the worst case. Therefore, by default, BinRank sets the *maxMSGSize* parameter to $(\mu + \sigma)$, and recomputes bins for all the MSGs with higher link counts, using the halved *maxBinSize*, as described in Section 3.6. Recall from Figure 3.6 that reducing the *maxBinSize* linearly reduces the query time, thus dramatically reducing the number of queries running over budget.

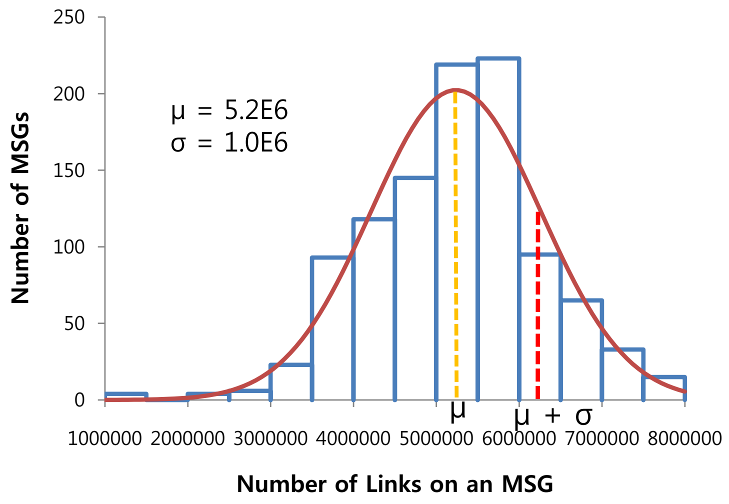


Figure 3.14: The distribution of the number of links on an MSG (1043 MSGs generated by using *maxBinSize* = 4000 and ϵ=5.0E-4)

In general case, *maxMSGSize* could be set to $(\mu + x\sigma)$, where good candidates for x are within $[0, 1]$ as we can see in our experimental results. In future, we plan to investigate optimizing x, while considering such factors as the time and space budget for MSG generation. For example, if $x = 0$, we need to regenerate about 50% of the MSGs, while we regenerate only 14% of them when $x = 1$.

Another approach we are planning to investigate, is to base *maxMSGSize* on the actual query performance measurements. The BinRank running time also follows a normal distribution $N(\mu_t, \sigma_t^2)$ and the time budget, 1 second in our experiments, corresponds to $\mu_t + 0.58\sigma_t$. Since the BinRank running time and the number of links on an MSG are highly correlated as shown in Figure 3.13, we can use 0.58 as x to select MSGs to regenerate.

3.8.7 Performance Comparison of BinRank with Monte Carlo Method and HubRank

In this section, we present a performance comparison of BinRank over Monte Carlo style methods and HubRank. We implemented the Monte Carlo algorithm 4, "MC complete path stopping at dangling nodes", introduced in [ALNO07] and HubRank[Cha07] that combines a hub based approach and a Monte Carlo method called fingerprint.

For a given keyword query, the Monte Carlo algorithm simulates random walks starting from nodes containing the keyword. Within a specified number of walks, it samples exactly the same number of random walks per each starting point. The authority score of a node is the total number of visits to the node divided by the total number of visits. Figure 3.15 shows the performance of the Monte Carlo algorithm in terms of accuracy of top-k lists and various query times. We used our workload keyword queries, and executed the Monte Carlo algorithm with different total numbers of sampled walks. As the number of sampled walks increases, the algorithm generates higher quality top-k lists, which usually takes more time.

However, we can see that τ values in Figure 3.15 are not as high as those of BinRank in Figure 3.7. With *maxBinSize* = 2000 or 4000 and ϵ =5E-4, BinRank generates high quality top-k lists of $\tau \approx 0.95$ in 350-750ms on average as shown in Figure 3.7 and Figure 3.9. However, according to Figure 3.15, the Monte Carlo algorithm generates top-k lists of $\tau \approx 0.70$ within the same amount of time. To get high quality top-k lists, it would take the Monte Carlo algorithm around 7 seconds per query term, which is probably not acceptable in a online search system.

We also implemented HubRank[Cha07] in order to measure the performance and the top-k quality over G_{wiki}. We selected hubs and then materialized a large number of fingerprints, while keeping the hub set fairly focused to our experimental query workload to save preprocessing cost. For a given keyword query, HubRank generates the active graph of the query by expanding the baseset's neighborhood until bounded by hub nodes or nodes very far from the given query node. Since G_{wiki} contains 3.2M nodes and 109M links, we often needed to compute many

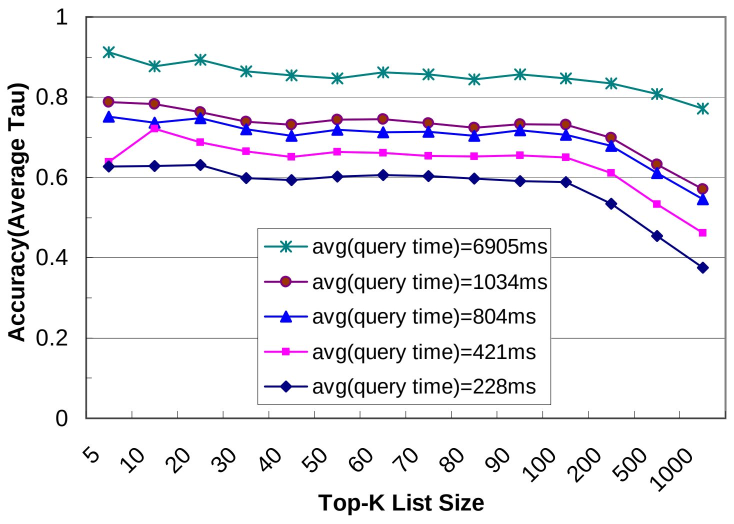


Figure 3.15: top-k accuracy of Monte Carlo algorithm with various query times

(thousands) of active vectors to answer a single query, where each active vector is a (sparse) vector of 3.2 million numbers. Due to this requirement, for most queries, we could not keep all the necessary active vectors in memory. The authors of [Cha07] also reported, that their implementation ran out of memory on a few queries, while they were running the experiments on a graph with less than a million edges. A two orders of magnitude increase in the size of the graph made this problem ubiquitous and prevented us from obtaining comparable results.

3.9 Conclusions

Summary. In this chapter, we proposed BinRank as a practical solution for scalable dynamic authority-based ranking. It is based on partitioning and approximation using a number of materialized subgraphs. We showed that our tunable system offers a nice trade off between query time and preprocessing cost.

We introduce a greedy algorithm that groups co-occurring terms into a number of bins for which we compute materialized subgraphs. Note that the number of bins is much less than the number of terms. The materialized subgraphs are computed off-line by using ObjectRank itself. The intuition behind the approach is that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms. Our extensive experimental evaluation confirms this intuition.

Future Work. For future work, we want to study the impact of other keyword relevance measures, besides term co-occurrence, such as thesaurus or ontologies, on the performance of BinRank. By increasing the relevance of keywords in a bin, we expect the quality of materialized subgraphs, thus the top-k quality and the query time, can be improved.

We also want to study better solutions for queries whose random surfer starting points are provided by boolean conditions. And ultimately, although our system is tunable, the configuration of our system ranging from number of bins, size of bins, tuning of the ObjectRank algorithm itself (edge weights, thresholds) is quite challenging, and a wizard to aid users is desirable.

To further improve the performance of BinRank, we plan to integrate BinRank and HubRank[Cha07] by executing HubRank on MSGs BinRank generates. Currently, we use the ObjectRank algorithm on MSGs in query time. Even though HubRank is not as scalable as BinRank, it performs better than ObjectRank on smaller graphs such as MSGs. In this way, we can leverage the synergy between BinRank and HubRank.

Chapter 3 was published in Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE-2009) and will appear in IEEE Transactions on Knowledge and Data Engineering 2010, Special Issue on the Best Papers of ICDE09 (TKDE-2010). Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp, “BinRank: Scaling Dynamic Authority-Based Search Using Materialized SubGraphs” and its extended version. The dissertation author was the primary investigator and author of these papers.

Chapter 4

Inverse ObjectRank: Measuring Specificity

4.1 Introduction

Even though ObjectRank [BHP04] is an effective search method generating high recall search results(Section 2.5), ranking solely by ObjectRank can be problematic, since general-content nodes may be ranked higher than nodes with content specific to the query. For example, consider the publications database of Figure 4.1, where edges denote citations (edges start from citing and end at cited paper), and the keyword query "Sorting". Then, using ObjectRank the "Access Path Selection in a Relational Database Management System" paper would be ranked highest, because it is cited by four papers containing "sorting" (or "sort"). The "Fundamental Techniques for Order Optimization" paper would be ranked second, since it is cited by only three "sorting" papers. This is unintuitive since the "Access Path Selection" paper has general content while the "Fundamental Techniques for Order Optimization" paper is more focused (specific). The latter paper should be ranked higher because it is mostly cited by "sorting" papers, whereas the former paper is also cited by many (the three papers on the top right) papers irrelevant to "sorting". This lack of specificity can also be viewed as a topic-drift problem [BH98, CJT01].

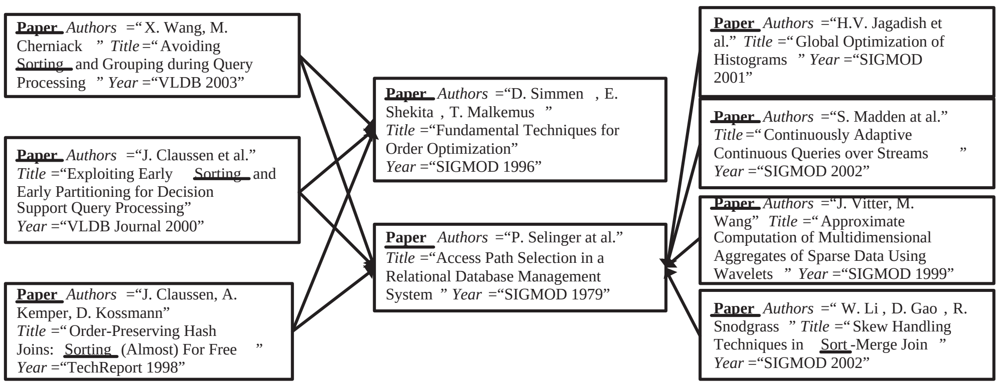

Paper Authors =“X. Wang, M. Cherniack ” Title =“Avoiding Sorting and Grouping during Query Processing ” Year =“VLDB 2003”
Paper Authors =“J. Claussen et al.” Title =“Exploiting Early Sorting and Early Partitioning for Decision Support Query Processing” Year =“VLDB Journal 2000”
Paper Authors =“J. Claussen, A. Kemper, D. Kossmann” Title =“Order-Preserving Hash Joins: Sorting (Almost) For Free ” Year =“TechReport 1998”
Paper Authors =“D. Simmen , E. Shekita , T. Malkemus ” Title =“Fundamental Techniques for Order Optimization” Year =“SIGMOD 1996”
Paper Authors =“P. Selinger at al.” Title =“Access Path Selection in a Relational Database Management System ” Year =“SIGMOD 1979”
Paper Authors =“H.V. Jagadish et al.” Title =“Global Optimization of Histograms ” Year =“SIGMOD 2001”
Paper Authors =“S. Madden at al.” Title =“Continuously Adaptive Continuous Queries over Streams ” Year =“SIGMOD 2002”
Paper Authors =“J. Vitter, M. Wang” Title =“Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets ” Year =“SIGMOD 1999”
Paper Authors =“ W. Li , D. Gao , R. Snodgrass ” Title =“Skew Handling Techniques in Sort-Merge Join ” Year =“SIGMOD 2002”

Figure 4.1: Instance of a Publications Database

In this chapter, we present Inverse ObjectRank [HHP06, HHP08], a keyword-specific metric of specificity, based on the link structure of the data graph. In particular, given a keyword w, the Inverse ObjectRank score $p^w(v)$ of node v shows how specific v is with respect to w. In terms of the random surfer model, $p^w(v)$ is the probability that starting from v and following the edges on the opposite direction we are on a node containing w at a specific point in time. As is the case for ObjectRank, the random surfer at any time step may get bored and go back to v.

Google uses (to the best of our knowledge) IR techniques based on the content of the Web pages (e.g., document length), which ignore the link structure of the labeled graph (i.e., the Web). Clearly, IR specificity metrics (e.g., document length) are not adequate since a longer document may be more specific than a shorter one for a particular query. However, IR metrics can be used in conjunction to Inverse ObjectRank to measure specificity.

The semantic contribution of this thesis to the quality of authority-based keyword search is evaluated using two user surveys in Section 4.5. We have implemented a web interface[1] to query the DBLP database (with additional link information extracted from CiteSeer[2]). It allows users to combine three link-based

[1]available at `http://www.db.ucsd.edu/ObjectRank` and `http://dbir.cis.fiu.edu/BibObjectRank/`

[2]citeseer.ist.psu.edu/

ranking measures, ObjectRank [BHP04], Global ObjectRank [BHP04], and Inverse ObjectRank, in various ways by adjusting a set of calibrating parameters. The performance contribution of our work, BinRank [HBRN09, HBRN10], is presented and evaluated in Chapter 3.

Upon the essential formal background on authority search and ObjectRank [BHP04] in Section 2.1, Section 4.2 introduces the keyword search problem this chapter addresses. Then, in Section 4.3, we present the semantics of Inverse ObjectRank, our novel specificity metric, as well as ways to combine it with ObjectRank. Section 4.4 describes the system's architecture and the online demo. We present the results of two user surveys in Section 4.5 to demonstrate the semantic contribution of this chapter. Furthermore, related work is discussed in Section 4.6. Finally, we conclude in Section 4.7.

4.2 Keyword Search and Ranking Factors

In this section, the definition of the keyword search problem we address in [HHP06, HHP08] is provided with the outline of the ranking measures. To improve the effectiveness of link-based keyword search, we combine our new specificity measure, Inverse ObjectRank, with existing ranking measures introduced in the ObjectRank framework [BHP04].

A *keyword query* q is defined as a set of keywords. The result of a keyword query is a list of objects of the database (i.e., nodes of the data graph), ranked according to the query. Our ranking system in [HHP06, HHP08] ranks objects according to three desired properties listed below. Notice that there is other non-link-based factors (e.g., IR score of individual nodes [HGP03]) that can be incorporated in the ranking as well, but they are beyond the scope of our work.

First, we give an example that highlights the effectiveness of ranking measures we combine in [HHP06, HHP08].

Example 4.2.1. *Given keyword query "sort" on the data graph of Figure 4.1 with the authority transfer rates of Figure 2.4 and damping factor* $d = 0.85$*, a possible result is*

1. *Authors=D. Simmen, E. Shekita, T. Malkemus. Title="Fundamental Techniques for Order Optimization". Year=SIGMOD 1996*
2. *Authors=P. Selinger at al. Title="Access Path Selection in a Relational Database Management System". Year=SIGMOD 1979*
3. *Authors=X. Wang, M. Cherniack. Title="Avoiding Sorting and Grouping during Query Processing". Year=VLDB 2003*
4. *Authors=J. Claussen et al. Title="Exploiting Early Sorting and Early Partitioning for Decision Support Query Processing". Year=VLDB Journal 2000*
5. *Authors=J. Claussen, A. Kemper, D. Kossmann. Title="Order-Preserving Hash Joins: Sorting (Almost) For Free". Year=TechReport 1998*
6. *Authors=W. Li, D. Gao, R. Snodgrass. Title="Skew Handling Techniques in Sort-Merge Join". Year=SIGMOD 2002*

*Top result is ranked highest because it satisfies all three properties, even though it does not contain the given query "sort" in its title: It is **relevant** to the query as three "sort" papers cite it, **of high-quality** since it is cited by three papers, and **specific** as only papers about "sort" cite it.*

Now, we outline the three ranking factors that measure different aspects of a dataset by exploiting the link structure of the data graph modeling the dataset. A user can combine them in various ways to obtain effective link-based keyword search results.

Relevance to Query: ObjectRank [BHP04]
Results that either contain the keywords of the query or are semantically associated to the keywords of the query should be ranked higher. The latter factor is equivalent to being connected through paths on the data graph in the ObjectRank data model, where edges correspond to semantic associations. In [BHP04], the link-based relevance of a node v to a query w (assume a single-keyword query for now) is the ObjectRank value $r^w(v)$ of v discussed in Section 2.4.1.

Global quality: Global ObjectRank [BP98, GSBS03, BHP04]
Results of high quality should be ranked higher. The link structure of the data

graph is used to measure quality. In particular, nodes with high incoming authority flow are assumed to have higher quality. For example, a highly referenced paper should be ranked higher than a non-referenced paper if the other ranking properties are equal. In [BHP04], Global ObjectRank (defined in Section 2.4.1) is used, which is an effective link-based metric to measure the global authority, that is, the quality of a node of the data graph. The Global ObjectRank $r^G(u)$ of a node u is defined as the probability that a random surfer starting from any node of the authority transfer graph will be at u at a specific time. For the case of the Web, Global ObjectRank is equivalent to PageRank [BP98], whose value has been proven by the success of Google[3].

Specificity: Inverse ObjectRank [HHP06, HHP08]
Specific results (nodes) should be ranked higher. That is, results with content particular to the query are preferred over results with content that spans across many topics. Previous work has not considered any link-based specificity metric. In Section 4.3.1 we present and discuss in detail Inverse ObjectRank.

Notice that these three properties correspond to the specificity, keyword proximity and hyperlink awareness properties respectively, defined in XRANK [GSBS03]. The same three properties (although not explicitly enumerated) have been used in other works as well (e.g., [BNH$^+$02]).

4.3 Inverse ObjectRank

In this section, we formally define our new ranking metric, Inverse ObjectRank [HHP06, HHP08], that measures the specificity of search results. To produce effective search results for a given keyword query, users can combine Inverse ObjectRank with existing metrics, ObjectRank and Global ObjectRank, discussed in Section 2.4.1. Finally, in Section 4.3.3 we present and address the challenges in combining these metrics into a ranking function.

[3]http://www.google.com

4.3.1 Inverse ObjectRank

Before presenting the specifics of Inverse ObjectRank, we explain why the traditional IR specificity metrics are inadequate. In particular, IR metrics ignore the link structure which makes them incomplete. For example, the document length (dl) metric cannot distinguish between objects (nodes) of approximately the same length, as is the case in our bibliographic database of paper titles and author names. Traditional IR specificity metrics are complementary to Inverse ObjectRank since they focus on the nodes of the authority flow graph, whereas Inverse ObjectRank exploits the edges. In this work we only evaluate Inverse ObjectRank and other alternative link-structure based specificity metrics in Section 4.5.1.

The intuition behind Inverse ObjectRank is the following. Given a keyword w, the ObjectRank value of a node v is the probability that a random surfer starting from a node containing w will be at v at a specific time. v is *specific* with respect to w if there is only few such keywords for which a surfer will end up on v starting from them. That is, if the random surfer will start at v and follow the edges of the authority transfer graph on the reverse direction, he/she should land back on w with high probability.

The above intuition is formally defined as follows. We first need to define the *inverse authority transfer graph* $D^I(V_D, E_D^I)$, given the authority transfer data graph $D^A(V_D, E_D^A)$, as follows: For every edge $e(u \to v) \in E_D^A$, we create an opposite-direction edge $e^I(v \to u) \in E_D^I$ with authority flow rate $a(e^I) = a(e)\frac{OutDeg(u)}{InDeg(v)}$. Notice that $1/OutDeg(u)$ is used in the calculation of $a(e)$, so by multiplying by $OutDeg(u)$ this is evened out.

Given a single-keyword query $q = \{w\}$, the Inverse ObjectRank score $p^w(u)$ of a node u is the probability that a random surfer of the inverse authority transfer graph D^I starting from u will be at a node containing w at a specific time.

Inverse ObjectRank is calculated in two steps. First, for each node $v \in D^I$ we compute its *connectivity*[4] $q^u(v)$ to u, i.e., how much authority starting from u

[4]This could also be called Inverse ObjectRank with respect to u. However, we avoid using this name which we reserve for the product of the final (second) step of the computation.

will reach v through D^I.

$$\mathbf{q}^u = d\mathbf{A}^I\mathbf{q}^u + (1-d)\mathbf{s}_u \tag{4.1}$$

where A^I is the transition matrix of D^I. That is, $A^I_{ij} = \alpha(e)$ if there is an edge $e = (v_j \rightarrow v_i)$ in D^I and 0 otherwise. $\mathbf{s}_u = [s_{u1}, \ldots, s_{un}]^T$ is the base set vector containing just u, i.e., $s_{ui} = 1$ if v_i is u and $s_{ui} = 0$ otherwise. Note that the connectivity $q^u(v)$ of a node v is equivalent to the inverse P-distance from u to v as defined by Jeh and Widom [JW03].

Second, the Inverse ObjectRank $p^w(u)$ is computed by summing the connectivities $q^u(v)$ of all nodes that contain w. That is

$$p^w(u) = \sum_{v \in S(w)} q^u(v) \tag{4.2}$$

where $S(w)$ is the base set of w as defined in Equation (2.5).

Global Inverse ObjectRank $\mathbf{p}$, which we do not use in our ranking function but has its own merit, is calculated by Equation 4.3. High Global Inverse ObjectRank denotes high connectivity of a node in a way similar to hub nodes in [Kle99].

$$\mathbf{p} = d\mathbf{A}^I\mathbf{p} + \frac{1-d}{|V|}\mathbf{e} \tag{4.3}$$

where $\mathbf{e} = [1, \ldots, 1]^T$.

Notice that Inverse ObjectRank is a keyword-specific metric of specificity, in the same sense that ObjectRank is a keyword-specific metric of relevance. This is the key reason why it performs superior to keyword-independent specificity heuristic metrics (including Global Inverse ObjectRank) as we show in Section 4.5. Also notice that Inverse ObjectRank has the same convergence properties as ObjectRank, which are described in Section 2.1.

4.3.2 Parallelisms to Information Retrieval Factors

Information Retrieval is a mature area which traditionally tackles the problem of ranking a set of documents with respect to a (typically keyword) query. On the other hand, keyword search [BNH+02, ACD02, HP02, GSBS03, GSVGM98] in data graphs and especially link-based keyword search [BP98, Hav02, Kle99] are young research areas. In this section we discuss how the basic IR factors (in particular, term weighting factors) correspond to properties of the data graph. In particular, the most widely accepted metrics to rank text documents for a keyword query are (a) the term frequency (tf), (b) the inverse document frequency (idf), and (c) the document length (dl).

ObjectRank corresponds to the $tf \cdot idf$ factor because if many nodes (similarly to tf) containing a keyword point to a node u, then the ObjectRank value of u increases, and if few nodes are in the base set, then they have higher weight (similarly to idf). On the other hand, Inverse ObjectRank corresponds to tf/dl because if we imagine that the node is expanded to a supernode following the incoming edges, then Inverse ObjectRank is proportional to the ratio of nodes in this supernode that contain the keywords.

4.3.3 Combine Ranking Factors and Multiple Keywords

There are two levels of combining scores in our framework to reach a ranking function for node v given a multiple-keyword query "$q = \{w_1, \ldots, w_m\}$". First, we need to find the score $f^{w_i}(v)$ ($f^{w_i}(v)$ is the score of node v given keyword w_i) of v for every single keyword w_i, and then combine these scores (and possibly Global ObjectRank $r^G(v)$) to compute the final score $f^q(v)$.

First, we define two alternative ways to combine ObjectRank with Inverse ObjectRank to compute $f^{w_i}(v)$, shown in Equations 4.4 and 4.5. The two equations are used to boost or downplay the weight of Inverse ObjectRank, that is, of the specificity factor in a keyword query respectively.

$$f^{w_i}(v) = r^{w_i}(v) \cdot p^{w_i}(v) \tag{4.4}$$

$$f^{w_i}(v) = r^{w_i}(v) \cdot \sqrt{p^{w_i}(v)} \tag{4.5}$$

Alternatively, if we choose a different specificity metric (see Section 4.5) we can replace $p^{w_i}(v)$ in Equation 4.4 by that metric, where we also show that Equation 4.5 typically produces superior results.

Second, we define the semantics of a multiple-keywords query "$q = \{w_1, \ldots, w_m\}$" by naturally extending the multiple-keywords random walk model. In particular, for the case of ObjectRank we consider m independent random surfers, where the ith surfer starts from the keyword base set $S(w_i)$. For AND semantics, the ObjectRank of an object v with respect to the m-keywords query is the probability that, at a given point in time, the m random surfers are simultaneously at v. We extend this model by substituting $r^{w_i}(v)$ by $f^{w_i}(v)$. Hence the score $f^q(v)$ of node v with respect to the m keywords is

$$f^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} f^{w_i}(v). \tag{4.6}$$

For OR semantics, the ObjectRank of v is the probability that, at a given point in time, *at least one* of the m random surfers will reach v. Hence, for two keywords w_1 and w_2 the model can be extended to

$$f^{w_1,w_2}(v) = f^{w_1}(v) + f^{w_2}(v) - f^{w_1}(v)f^{w_2}(v) \tag{4.7}$$

and for more than two it is defined accordingly, as specified by the inclusion-exclusion principle (also known as the sieve principle). Notice that [Hav02] also takes the sum of the topic-sensitive PageRank values to calculate the PageRank of a page.

If Global ObjectRank is included in the computation, it is treated as an additional keyword w_{m+1} with $f^{w_{m+1}}(v) = r^G(v)$.

4.4 Architecture

We have implemented a system to answer keyword queries on databases. The user inputs (a) a set of keywords, (b) a choice for combining semantics (AND or OR), (c) the importance of global quality of the results (i.e., Global ObjectRank), (d) the importance of containing the actual query keywords (translated to a damping factor value d), and (e) a specificity metric (as we explain in Section 4.5). The output of the system is a ranked list of nodes of the database (to be more formal, of the authority transfer graph) according to the input parameters based on the

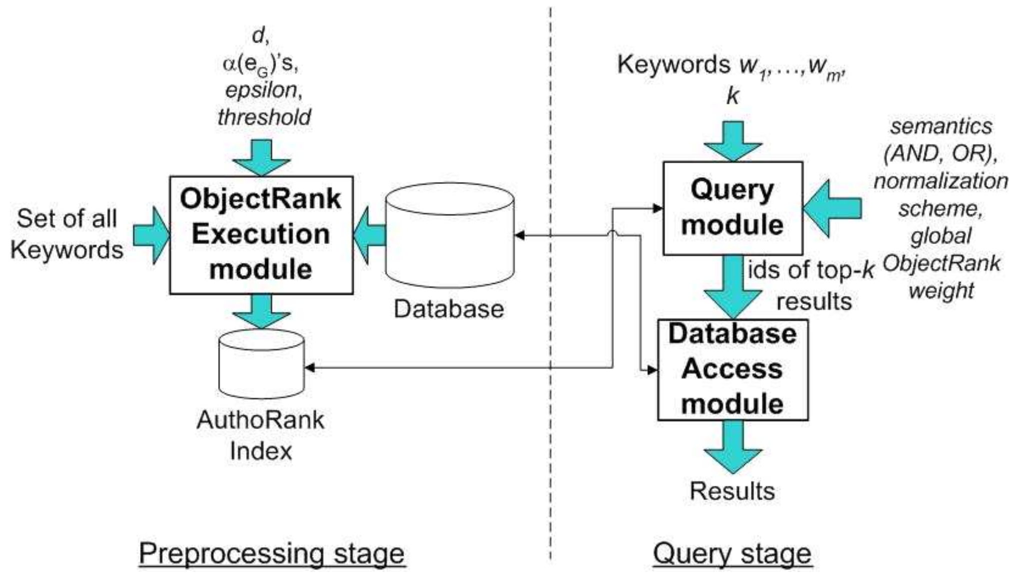


Figure 4.2: System Architecture.

ranking function in Equation 4.6 or 4.7 (for AND and OR semantics respectively). The authority transfer graph is stored in a relational database using the schema shown in Figure 2.3.

The architecture of the system, which is shown in Figure 4.2, is divided into two stages. The preprocessing stage consists of the *Authority Flow Execution module*, which inputs the authority transfer graph G to be indexed, the set of all keywords that will be indexed, and a set of parameters. In particular these parameters are: (i) A set of damping factors d, that users are expected to choose from. (ii) The convergence constant *epsilon* which determines when the ObjectRank and Inverse ObjectRank algorithms converge, and (iii) The *threshold* value which determines the minimum score that an object must have to be stored in the authority flow index. Note that other index pruning techniques are possible [CCF+01]; however, we found that this simple uniform pruning technique performs well in our setting.

The Authority Flow Execution module creates the *authority flow index*, which is an inverted index, indexed by the keywords. For each keyword w, it stores a list of $\langle id(u), f^w(u)\rangle$ pairs for each object u that has $f^w(u) \geq threshold$. The pairs are sorted by descending $f^w(u)$ to facilitate an efficient querying method as we describe below. The authority flow index has been implemented as an index-based table, where the lists are stored in a CLOB attribute. A hash-index is built on top of each list to allow random access, which is required by the Query module. Note that if we allow multiple combinations of calibration parameters to be selected by the user, then we create multiple inverted indexes, one for each such combination.

The *Query module* inputs a set of keywords $w_1, \ldots, w_m$ and a set of adjusting parameters, and outputs the top-k objects according to the ranking function (Equation 4.6 or 4.7). In particular, these parameters are: (a) a choice for combining semantics (AND or OR), (b) the importance of global quality of the results (i.e., Global ObjectRank), (c) the importance of containing the actual query keywords (translated to a damping factor value d), and (d) a specificity metric (as we explain in Section 4.5). The keyword-specific lists read from the authority flow index are merged using the *Threshold Algorithm* [FLN01] which is guaranteed to read the minimum prefix of each list. Notice that the Threshold Algorithm is applicable since both combining functions (Equations 4.6 and 4.7) are monotone.

Finally, the *Database Access module* inputs the result *id*s and queries the database to get the corresponding node of the authority transfer graph. This information is stored into an id-indexed table, that contains a CLOB attribute value for each object id. For example, a paper object CLOB would contain the paper title, the authors' names, and the conference name and year.

4.4.1 Demo

We have built a demo [HHP06] on bibliographic data, which is available online at two mirror sites[5]. The data was collected using the following method. First, we downloaded all publications and citations from the DBLP database[6]. We noticed that this source is missing too many citations, which greatly degrades the quality of link-based analysis. To overcome this shortcoming, we used Citeseer[7] as an additional citations' source. We built a web crawler to retrieve these citations

[5]http://www.db.ucsd.edu/ObjectRank/ and http://dbir.cis.fiu.edu/BibObjectRank/

[6]http://www.informatik.uni-trier.de/ ley/db/

[7]http://citeseer.ist.psu.edu/

since we found that the exported files of Citeseer are in a large degree inaccurate. We matched papers from the two sources using their titles, which of course can lead to few inaccurate matches.

Our demo offers to the user multiple authority flow settings, in order to accommodate multiple user profiles/requirements. We believe the ability to customize authority flow schemes is important, since we should not assume that "one size fits all" when it comes to opinions about authority flow. For example, there is one setting for users that primarily care for papers with high global importance and another for users that primarily care for papers that are directly or indirectly heavily referenced by papers that have the keywords. We expect that multiple settings make sense in all non-trivial applications.

4.5 Qualitative Evaluation

The user survey investigates and compares alternative ways to incorporate link-based specificity to keyword queries. In particular, we propose alternative specificity metrics and also experiment with various ways to incorporate Inverse ObjectRank in the ranking. We performed three qualitative experiments to compare these alternatives: a comparison to a textbook's bibliography, a user survey, and a quantitative measurement of the distances between the result lists. The key conclusion from these studies is that combining ObjectRank with the square root of Inverse ObjectRank produces the best results.

4.5.1 Combining Specificity with Relevance and Global Importance

We consider the following ranking functions. For each case we specify the single keyword score $f^{w_i}(v)$ of node v as well as the multiple keywords combining function $f^{w_1,\dots,w_m}(v)$. Notice that AND semantics is used.

1. *Obj* ranks according to ObjectRank. $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\dots,w_m}(v)$ is defined by Equation 4.6.

	Obj		*ObjInv*		*ObjOverGlobal*		*Objd03*		*ObjSqrtInv*	
	A-S	A-NS	A-S	A-NS	A-S	A-NS	A-S	A-NS	A-S	A-NS
tree index	7	1	6	1	0	0	6	1	7	1
hash index	3	3	1	0	0	0	0	0	2	1
concurrency control	4	2	7	0	0	0	7	1	7	1
object databases	1	4	3	0	0	0	4	2	4	1
deductive databases	4	2	4	0	0	0	4	0	5	0
spatial databases	3	2	1	0	0	0	2	0	2	0
distributed databases	1	3	5	0	0	0	5	1	6	1
relational model	3	5	3	2	0	0	3	2	3	4
query optimization	2	3	3	1	0	0	4	2	4	2
data mining	4	1	6	0	0	0	4	0	6	0
relational algebra	3	2	2	0	0	0	3	0	2	0
AVERAGE	3.18	2.55	3.73	0.36	0	0	3.82	0.82	4.36	1

Figure 4.3: Number of Authoritative-Specific and Authoritative-Non-Specific papers according to [RG03].

2. *ObjInv* ranks according to the product of ObjectRank and Inverse ObjectRank. $f^{w_i}(v)$ is defined by Equation 4.4 and $f^{w_1,\dots,w_m}(v)$ by Equation 4.6.

3. *ObjOverGlobal* uses the inverse of Global ObjectRank as the specificity metric. The assumption is that if a node has high ObjectRank, it receives it from a wide range of nodes, and hence this node is too general. It is $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\dots,w_m}(v) = \prod_{i=1,\dots,m} f^{w_i}(v)/r^G(v)$

4. *Objd03* is the same as *Obj* but $d = 0.3$ ($d = 0.85$ when not specified). That is, this ranking attempts to achieve specificity by limiting the authority flow and emphasizing the nodes that contain the keywords.

5. *ObjSqrtInv* ranks according to the product of ObjectRank and the square root of Inverse ObjectRank. $f^{w_i}(v)$ is defined by Equation 4.5 and $f^{w_1,\dots,w_m}(v)$ by Equation 4.6.

6. *ObjOverInc* uses the inverse of the number of incoming links $NumIncLinks(v)$ of node v as specificity metric. It is $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\dots,w_m}(v) = \prod_{i=1,\dots,m} f^{w_i}(v)/NumIncLinks(v)$. $NumIncLinks(v)$ can be viewed as an approximation of $r^G(v)$, so this ranking can be viewed as an approximation of *ObjOverGlobal*.

7. *ObjOverInvGlobal* uses the inverse of Global Inverse ObjectRank $r^{IG}(v)$ as the specificity metric. It is $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} f^{w_i}(v)/r^{IG}(v)$.

ObjOverInc and *ObjOverInvGlobal* were found to perform much worse than the other ranking functions and their results are omitted for simplicity.

4.5.2 Comparison to Textbook's Bibliography

We assume that the bibliography section of each chapter in [RG03] is a highly credible source of references related to the chapter title. Based on this assumption, we compare the recall (precision is the same as recall in this case) of the top-10 papers produced by the five above ranking functions with respect to the papers in the bibliography section of the corresponding chapter, which is viewed as the ground truth.

We evaluated 11 queries that correspond to chapter titles of the textbook [RG03]. For each keyword query q, let $B(q)$ denote the set of papers in the bibliography of the corresponding chapter and $U(q)$ denote the set of papers that are in the bibliography of the book but not of that chapter, that is, they are not in $B(q)$. We assume that papers in $B(q)$ satisfy all properties of Section 4.2, that is, they are specific to q, relevant to q and of high quality. We refer to such papers as authoritative-specific for q. On the other hand, papers in $U(q)$ have high quality but are not highly relevant or specific to q, and are referred to as authoritative-non-specific. Figure 4.3 shows the number of authoritative-specific and authoritative-non-specific papers for each query for the five ranking functions.

Obviously, *ObjOverGlobal* has the worst performance according to Figure 4.3. In particular, it produces no authoritative-specific or authoritative-non-specific papers in the top-10 results for any query. Hence, we do not consider this metric in our discussion henceforth. *Objd03*, which promotes papers that contain the actual keywords, performs well in terms of authoritative-specific results. The reason is that because the queries in Figure 4.3 refer to fundamental areas, it happens that many important papers contain the actual keywords.

Now, let's focus on the relationship between *Obj*, *ObjInv*, and *ObjSqrtInv*, which have the common property that they only involve keyword-specific computations. In terms of the number of authoritative-non-specific papers, *Obj* and *ObjInv* are located at the two extremes. We introduced *ObjSqrtInv* as a ranking function to combine the desirable properties of both ends. As expected, *ObjSqrtInv* has a number of authoritative-non-specific papers that is between those of *Obj* and *ObjInv*. However, *ObjSqrtInv* is superior than both *Obj* and *ObjInv* in terms of average number of authoritative-specific papers, which is a highly desirable property.

The intuition behind the selection of *ObjSqrtInv* is the following. Using *ObjInv*, a too specific object may receive a high score even if it has relatively low quality and relevance. For example, a very high quality object that happens to be relevant to 10 keywords would be ranked equal to a 10 times lower-quality document that is relevant to only one keyword. Hence, taking the square root of Inverse ObjectRank serves a purpose similar to taking the logarithm of *tf* in IR to avoid assigning top score to documents that repeat many times the keywords in an adversary way. We chose square root instead of logarithm because logarithm is sensitive to the breadth of the range of the Inverse ObjectRank values. In particular, we observed that few nodes have very large Inverse ObjectRank values which have orders of magnitude difference to the top ObjectRank values. Square root is more appropriate since $\sqrt{a \cdot c}/\sqrt{b \cdot c}$ does not depend on c ($c > 0$), whereas $\log(a \cdot c)/\log(b \cdot c)$ depends on c.

On the other hand, taking the square root of ObjectRank is a bad idea, since ObjectRank is the relevance (and quality) measure, which is the primary ranking factor, and cannot be easily tricked (especially in controlled databases like bibliographical). Other ways to decrease the weight of Inverse ObjectRank were tested, like dividing (1-d) by a constant in Equation 4.1, but taking the square root was found to perform better.

A surprising fact is that the average number of authoritative-specific papers for *Obj* is high. The reason is that the textbook contains multiple general references for each chapter, to introduce the topic to newcomers or carry very general

Obj	*ObjInv*	*ObjOverGlobal*	*Objd03*	*ObjSqrtInv*
2.13	3.42	2.13	3.60	3.92

Figure 4.4: Average Ratings of the Five Specificity Metrics at the User Survey.

concepts, which would not be judged as specific by an experienced researcher. This observation is also supported by the user survey presented below.

4.5.3 User Survey

We asked twelve users (not involved in the project), eight database professors and four database Ph.D. students in eight different universities in the US and abroad, to rank the top-10 result lists for the five ranking functions, for various queries. The survey consisted of 9 queries, 4 of which were chapter titles of [RG03]. Each user/subject assigned a score between 1 and 5 to each result list for the queries/topics he/she feels comfortable with. Also, the user can specify his/her level of expertise for each topic, which is then used to weight the rating when computing average numbers. We explained to the users what is meant by authoritative-specific as opposed to authoritative-non-specific by providing the following scenario, and we asked them to evaluate according to the former.

Survey Scenario: *Let us assume you are a professor and you need to give a reading list to a first year graduate student who starts research on a topic, say "XML database storage". Being a first year student, he/she likely has no background knowledge on database issues pertaining to XML and semistructured data in general. In this case, you may want to provide an authoritative papers list where it is OK (indeed desirable) to include a few seminal papers on XML and semistructured databases, even though they may not be related to storage in particular. Such seminal papers are a good starting point for the student. These papers are authoritative-non-specific papers. Instead, our survey asks for authoritative-specific papers. Now assume that you produce a reading list for someone (perhaps yourself)*

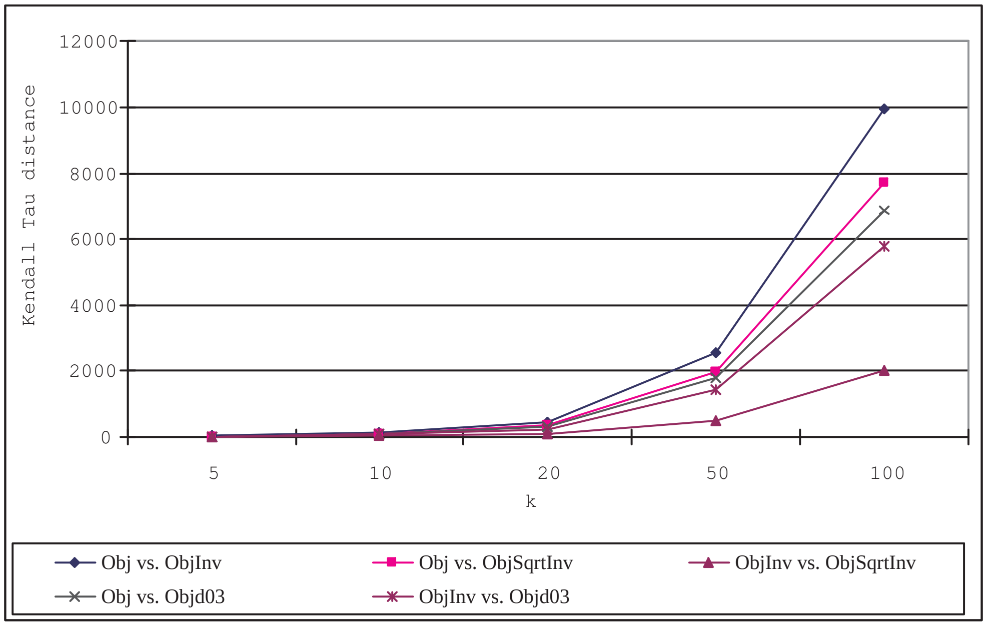


Figure 4.5: Compare Results' Distances.

who already knows the basics of XML databases and of conventional (relational) storage systems. You now care about the specific papers in XML storage, in particular.

The average ratings are shown in Figure 4.4. We observe that *ObjSqrtInv* has the highest average rating, which is consistent with our expectation that *ObjSqrtInv* outperforms other metrics because of its balance between authority and specificity. We also see that *Obj*, which lacks a specificity factor, received low ratings in contrast to Figure 4.3, where it received a high score due to the reasons mentioned above.

Surprisingly, *Objd03* received a high average rating, although setting $d = 0.3$ greatly degrades the authority flow factor and promotes results that contain the actual keywords. The reason of the high average rating is that some subjects did not have knowledge of the best papers for a topic and instead they seem to have judged by the titles of the papers and the presence of the keywords in them.

4.5.4 Distance Between Specificity Metrics

In this experiment, we perform a quantitative comparison between the above ranking functions using the Kendall Tau distances between the generated result lists. Since the two top-k lists are not permutations of each other, we use the extended Kendall Tau definition of Fagin et al. [FKS03]. The average Kendall Tau distances between the most interesting pairs of ranking functions over 100 queries are shown in Figure 4.5, as a function of the lists length k. Notice that as expected, there is a large distance between *Obj* and *ObjInv* but a smaller distance between *Obj* and *ObjSqrtInv*. We do not include the distance between *Obj* and *ObjOverGlobal* since their results are often disjoint hence resulting in very large distances.

4.6 Related Work

We first present how state-of-the-art works rank the results of a keyword query, using traditional IR techniques and exploiting the link structure of the data graph. Then we discuss about related work on the performance of link-based algorithms.

Traditional IR ranking. Currently, all major database vendors offer tools [Ora06, DB206, MSD06] for keyword search in single attributes of the database. That is, they assign a score to an attribute value according to its relevance to the keyword query. The score is calculated using well known ranking functions from the IR community [Sal89], although their precise formula is not disclosed. Recent works [BNH$^+$02, HP02, HPB03, ACD02] on keyword search on databases, where the result is a tree of objects, either use similar IR techniques [BNH$^+$02], or use the simpler boolean semantics [HP02, HPB03, ACD02], where the score of an attribute is 1 (0) if it contains (does not contain) the keywords.

The first shortcoming of these semantics is that they miss objects that are very related to the keywords, although they do not contain them (Section 2.2). The second shortcoming is that the traditional IR semantics are unable to meaningfully sort the resulting objects according to their relevance to the keywords. For example, for the query "XML", the paper [GNY+02] on Quality of Service that uses an XML-based language, would be ranked as high as a classic book on XML [ASB00]. Again, the relevance information is hidden in the link structure of the data graph.

As we discuss in Section 4.3.2, the most popular specificity metric in Information Retrieval is the document length (dl). As an example, a state-of-the-art IR ranking function is [Sin01]:

$$Score(a_i, Q) = \sum_{w \in Q \cap a_i} \frac{1 + ln(1 + ln(tf))}{(1-s) + s\frac{dl}{avdl}} \cdot ln\frac{N+1}{df} \tag{4.8}$$

where, for a word w, tf is the frequency of w in the document D, df is the number of documents in the database containing word w, dl is the size of D in characters, $avdl$ is the average document size, N is the total number of documents in the database, and s is a constant (usually 0.2). Croft [Cro00] and Craswell et al. [CRZT05] present techniques on combining ranking factors.

Link-based semantics. In [BHP04], Balmin et al. introduced the ObjectRank metric. This work completes [BHP04] in the following ways. The specificity factor is handled and evaluated, in contrast to [BHP04] where the specificity factor is ignored. Inverse ObjectRank is introduced and qualitatively evaluated. Furthermore, in this work we clearly identify the ranking factors (relevance, specificity and global importance) and map them to authority flow metrics. Moreover, we explain these authority flow metrics from the perspective of information theory. We also elaborate on the combining ranking function and study techniques to weigh the query keywords. Finally, we enriched the demo available on the Web by adding adjusting parameters, and including the whole DBLP dataset and citations from Citeseer, in contrast to [BHP04] where a small subset of DBLP was used.

Spreading activation techniques [Pre81, SB88] can be seen as earlier precedents of link-based semantics in that they performed search by controlling the propagation of node scores through associative networks. To the best of our knowledge, Savoy [Sav92] was the first to use the link structure of the Web to discover relevant pages. This idea became more popular with PageRank [BP98], where a global score is assigned to each Web page as we explain in Section 2.1. However, directly applying the PageRank approach in our problem is not suitable as we explain in Section 2.2. HITS [Kle99] employs mutually dependent computation of two values for each web page: hub value and authority. In contrast to PageRank, it is able to find relevant pages that do not contain the keyword, if they are directly pointed by pages that do. However, HITS does not consider domain-specific link semantics and does not make use of schema information. The relevance between two nodes in a data graph can also be viewed as the resistance between them in the corresponding electrical network, where a resistor is added on each edge. This approach is equivalent to the random walk model [DS84].

Richardson et al. [RD02] propose an improvement to PageRank extending the work of Bharat and Henzinger [BH98], where the random surfer takes into account the relevance of each page to the query when navigating from one page to the other. However, they require that every result contains the keyword, and ignore the case of multiple keywords. Haveliwala [Hav02] proposes a topic-sensitive PageRank, where the topic-specific PageRanks for each page are precomputed and the PageRank value of the most relevant topic is used for each query. Both works apply to the Web and do not address the unique characteristics of structured databases, as we discuss in Section 2.2. Furthermore, they offer no adjusting parameters to calibrate the system according to the specifics of an application.

Recently, the idea of PageRank has been applied to structured databases [GSBS03, HXY03]. XRANK [GSBS03] proposes a way to rank XML elements using the link structure of the database. Furthermore, they introduce a notion similar to our ObjectRank transfer edge bounds, to distinguish between containment and IDREF edges. Huang et al. [HXY03] propose a way to rank the tuples of a relational database using PageRank, where connections are determined dynamically by the query workload and not statically by the schema. However, none of these works exploits the link structure to provide keyword-specific ranking. Furthermore, they ignore the schema semantics when computing the scores.

Geerts et al. [GMT04] use a set of queries to rank the values of a relational database using authority flow semantics. TrustRank [GGMP04] uses the idea of

Global Inverse PageRank as a heuristic for a completely different purpose than specificity. In particular, they use it to find well connected pages to use as seeds in their algorithms.

Performance. A set of works [Hav99, CGS02, JW03, KHMG03] have tackled the problem of improving the performance of the original PageRank algorithm. [Hav99, CGS02] present algorithms to improve the calculation of a global PageRank. Jeh and Widom [JW03] present a method to efficiently calculate the PageRank values for multiple base sets, by precomputing a set of *partial vectors* which are used in runtime to calculate the PageRanks. The key idea is to precompute in a compact way the PageRank values for a set of hub pages, through which most of the random walks pass. Then using these hub PageRanks, calculate in runtime the PageRanks for any base set consisting of nodes in the hub set. However, in our case it is not possible to define a set of hub nodes, since any node of the database can be part of a base set.

4.7 Conclusions

In this chapter, we presented Inverse ObjectRank, which is a link-based and keyword-specific specificity metric. We showed how Inverse ObjectRank is combined with other ranking measures to produce the results list for a keyword query. Our methods have been qualitatively evaluated using a user survey and the bibliography sections of a database textbook. We concluded that combining ObjectRank with the square root of Inverse ObjectRank produces results of highest quality. Furthermore, we built a prototype of our methods on a bibliographic database, which we made available on the Web.

Chapter 4 was published in Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD-2006), pp 796-798 and ACM Transactions on Database Systems 2008, 33(1) (TODS-2008). Heasoo Hwang, Vagelis Hristidis, and Yannis Papakonstantinou, "ObjectRank: a system for authority-based search on databases" and Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou, “Authority-based keyword search in databases”. The dissertation author and Vagelis Hristidis were the primary investigators and authors of these papers.

Chapter 5

Summary and Conclusions

In this dissertation, we aimed at supporting efficient and effective keyword search functionality over graph-structured data by exploiting the semantic connections between data objects.

We first addressed the performance issue of dynamic authority-based ranking methods such as personalized PageRank and ObjectRank. To improve the query execution time required to compute query-specific relevance scores for top-K keyword search, we introduced a novel approach, BinRank, that approximates dynamic link-based ranking scores efficiently by using a number of materialized subgraphs. It partitions a dictionary into bins of relevant keywords and then constructs materialized subgraphs (MSGs) per bin in preprocessing stage. We suggested a greedy algorithm that groups co-occurring terms into a number of bins for which we compute materialized subgraphs. The materialized subgraphs are computed off-line by using ObjectRank itself. The intuition behind the approach is that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms. Our extensive experimental evaluation confirms this intuition. Since the number of bins is much less than the number of terms, the preprocessing stage of BinRank is performed efficiently. In query time, to produce highly accurate top-K results efficiently, BinRank uses the MSG corresponding to the given keyword, instead of the original data graph. We showed that our tunable system offers a nice trade-off between query time and preprocessing cost.

Then, we presented Inverse ObjectRank, which is a link-based and keyword-specific specificity metric. PageRank and ObjectRank calculate the global importance score and the query-specific authority score of each node respectively by exploiting the link structure of a given data graph. However, both measures favor nodes with high in-degree that may contain popular yet generic content, and thus those nodes are frequently included in top-K lists, regardless of given query. Inverse ObjectRank measures the content-specificity of each node by traversing the semantic links in the data graph in the reverse direction. We built a prototype of our methods on a bibliographic database, which we made available on the Web. We allowed users to adjust the importance of the three ranking measures (global importance, query-relevance, and content-specificity) to improve the quality of search results. We showed how Inverse ObjectRank is combined with other ranking measures to produce the results list for a keyword query. Our methods have been qualitatively evaluated using a user survey and the bibliography sections of a database textbook. Our experimental evaluation showed that combining ObjectRank with the square root of Inverse ObjectRank produces results of highest quality.

Bibliography

[ACD02] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. DBXplorer: A System For Keyword-Based Search Over Relational Databases. *ICDE*, 2002.

[ALNO07] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in PageRank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.*, 45(2):890–904, 2007.

[ASB00] Serge Abiteboul, Dan Suciu, and Peter Buneman. Data on the Web : From Relations to Semistructured Data and Xml. *Morgan Kaufmann Series in Data Management Systems*, 2000.

[BdJKT05] Jeremy T. Bradley, Douglas V. de Jager, William J. Knottenbelt, and Aleksandar Trifunovic. Hypergraph partitioning for faster parallel PageRank computation. In *EPEW/WS-FM*, pages 155–171, 2005.

[BH98] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR*, 1998.

[BHP04] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. ObjectRank: Authority-based keyword search in databases. In *VLDB*, 2004.

[BHR$^+$07] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD*, pages 199–210, 2007.

[BNH$^+$02] G. Bhalotia, C. Nakhey, A. Hulgeri, S. Chakrabarti, and S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. *ICDE*, 2002.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[CA06] Soumen Chakrabarti and Alekh Agarwal. Learning parameters in entity relationship graphs from ranking preferences. In *ECML/PKDD, volume 4213 of LNCS*, 2006.

[CCF$^+$01] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. Static index pruning for information retrieval systems. In *ACM SIGIR*, 2001.

[CDG$^+$98] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *WWW*, 1998.

[CGS02] Y. Chen, Q. Gan, and T. Suel. I/O-efficient techniques for computing PageRank. *CIKM*, 2002.

[Cha07] Soumen Chakrabarti. Dynamic personalized PageRank in entity-relation graphs. In *WWW*, 2007.

[CJT01] Soumen Chakrabarti, Mukul Joshi, and Vivek Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *SIGIR*, 2001.

[Cro00] W. Bruce Croft. Combining Approaches to Information Retrieval. *Advances in Information Retrieval: Recent Research from the CIIR, Kluwer, Chapter 1*, 2000.

[CRZT05] Nick Craswell, Stephen E. Robertson, Hugo Zaragoza, and Michael J. Taylor. Relevance weighting for query independent evidence. In *SIGIR*, 2005.

[CS07] Junghoo Cho and Uri Schonfeld. Rankmass crawler: A crawler with high PageRank coverage guarantee. In *VLDB*, 2007.

[DB206] http://www.ibm.com/software/data/db2/extenders/textinformation/index.html. 2006.

[DDF$^+$90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[DEGP98] S. Dar, G. Entin, S. Geva, and E. Palmon. DTL's DataSpot: Database Exploration Using Plain Language. *VLDB*, 1998.

[DS84] P. G. Doyle and J. L. Snell. Random Walks and Electric Networks. *Mathematical Association of America, Washington, D. C.*, 1984.

[FKS03] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Procs.ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal Aggregation Algorithms for Middleware. *ACM PODS*, 2001.

[FRCS05] Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[GGMP04] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen. Combating Web Spam with TrustRank. *VLDB*, 2004.

[GJ85] Michael R. Garey and David S. Johnson. A 71/60 theorem for bin packing. *Journal of Complexity*, 1:65106, 1985.

[GMT04] Floris Geerts, Heikki Mannila, and Evimaria Terzi. Relational link-based ranking. *VLDB*, 2004.

[GNY$^+$02] X. Gu, K. Nahrstedt, W. Yuan, D. Wichadakul, and D. Xu. An XML-based Quality of Service Enabling Language for the Web. *Journal of Visual Languages and Computing 13(1): 61-95*, 2002.

[GSBS03] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. *ACM SIGMOD*, 2003.

[GSVGM98] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity Search in Databases. *VLDB*, 1998.

[Hav99] Taher H. Haveliwala. Efficient computation of PageRank. *Technical report, Stanford University (http://www.stanford.edu/~taherh/papers/efficient-pr.pdf)*, 1999.

[Hav02] Taher H. Haveliwala. Topic-sensitive PageRank. In *WWW*, 2002.

[HBPR07] Heasoo Hwang, Andrey Balmin, Hamid Pirahesh, and Berthold Reinwald. Information discovery in loosely integrated data. In *SIGMOD*, 2007.

[HBRN09] Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp. BinRank: Scaling dynamic authority-based search using materialized subgraphs. In *ICDE*, 2009.

[HBRN10] Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp. BinRank: Scaling dynamic authority-based search using materialized subgraphs (extended version). *To appear in IEEE Transactions on Knowledge and Data Engineering (Speical Issue on the Best Papers of ICDE 2009)*, 2010.

[HGP03] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. *VLDB*, 2003.

[HHP06] Heasoo Hwang, Vagelis Hristidis, and Yannis Papakonstantinou. ObjectRank: a system for authority-based search on databases. In *SIGMOD*, 2006.

[HHP08] Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou. Authority-based keyword search in databases. *ACM Trans. Database Syst.*, 33(1), 2008.

[HP02] Vagelis Hristidis and Yannis Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. *VLDB*, 2002.

[HPB03] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. *ICDE*, 2003.

[HXY03] A. Huang, Q. Xue, and J. Yang. TupleRank and Implicit Relationship Discovery in Relational Databases. *WAIM*, 2003.

[JW03] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW*, 2003.

[Ken55] M.G. Kendall. *Rank Correlation Methods*. NewYork: Hafner Publishing Co., 1955.

[KHMG03] Sepandar Kamvar, Taher H. Haveliwala, Christopher Manning, and Gene Golub. Extrapolation Methods for Accelerating PageRank Computations. *WWW Conference*, 2003.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*, 1999.

[MCN06] Einat Minkov, William Cohen, and Andrew Ng. A graphical framework for contextual search and name disambiguation in email. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 1–8, 2006.

[Min07] Einat Minkov. Learning to rank typed graph walks: Local and global approaches. In *WebKDD/KDD-SNA workshop*, 2007.

[MR95] R. Motwani and P. Raghavan. Randomized Algorithms. *Cambridge University Press, United Kingdom*, 1995.

[MSD06] http://msdn.microsoft.com/library/. 2006.

[NZWM05] Zaiqing Nie, Yuanzhi Zhang, Ji-Rong Wen, and Wei-Ying Ma. Object-level ranking: bringing order to web objects. In *WWW*, pages 567–574, 2005.

[Ora06] http://technet.oracle.com/products/text/content.html. 2006.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. In *Technical report, Stanford University*, 1998.

[Pre81] Scott Everett Preece. *A spreading activation network model for information retrieval.* PhD thesis, Champaign, IL, USA, 1981.

[RD02] M. Richardson and P. Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. *Advances in Neural Information Processing Systems 14, MIT Press*, 2002.

[RG03] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems. Third Edition.* McGraw-Hill Book Co, 2003.

[Sal89] Gerard Salton. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. *Addison Wesley*, 1989.

[Sav92] Jacques Savoy. Bayesian inference networks and spreading activation in hypertext systems. *Information Processing and Management*, 28(3):389–406, 1992.

[SB88] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information. In *SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 147–160, New York, NY, USA, 1988. ACM.

[Sin01] Amit Singhal. Modern information retrieval: a brief overview. *IEEE Data Engineering Bulletin, Special Issue on Text and Databases*, 24(4), December 2001.

[SKI08] Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. Précis: from unstructured keywords as queries to structured databases as answers. *The VLDB Journal*, 17(1):117–149, 2008.