WORLD WIDE WEB RESOURCE DISCOVERY

The First Year Report

Submitted to the School of Applied Science

of the Nanyang Technological University

By

Xu Jian

In partial fulfillment of the requirements

for the degree of

Master of Applied Science

September 1998

Acknowledgements

First, I would like to express my deepest gratitude to my supervisor, Dr. Lim Ee Peng, not only for his invaluable guidance and constructive criticism throughout the research, but also for his encouragement and understanding. Also from him, I gained tremendously a lot on my research, writing and presentation skills. I thank him for teaching me about research, motivating me by his enthusiasm and optimism, and helping me to understand the process of accomplishing it.

I would like to thank my co-supervisor, Dr. Ng Wee Keong for leading me to this fascinating research field. I have learned from him many ways to do research, such as the way of describing a complex problem with very simple terms, etc.

Thanks also go to all the research students and technicians of Centre for Advanced Information Systems. In particular, I am thankful to Qin Fengqong, Yan Guanghao, Liu Haifeng and technician Tan Lay Choo for their great help to me on doing this research.

Finally, I would like to thank my parents and other family members for always giving me moral support. And specially thanks to my wife, Li Jinglin, for her encouragement and understanding.

Abstract

Query routing refers to the general problem of selecting from a large set of accessible information sources the ones relevant to a given query(i.e. **database selection**), evaluating the query on the selected sources(i.e. **query evaluation**), and merging their results(i.e. **result merging**). As the number of information sources on the Internet increases dramatically, query routing is becoming increasingly important. Much of the previous work in query routing focused on information sources that are document collections. Moreover, there has been little work done for collections that can be accessed only through some query interface. In this project, we focus on the database selection problem, an important subproblem of query routing, for bibliographic databases consisting of multiple text attributes. In particular, we first proposed a few database selection techniques that are designed for bibliographic databases accessible through some boolean retrieval interface. Our techniques rely on past queries and query results to determine the relevance of databases with respect to a given user query. By conducting a series of experiments on a set of bibliographic databases, we evaluated and compared the performance of our proposed database selection techniques. We further explore the use of techniques to improve the performance of database selection for bibliographic databases. By clustering bibliographic records of each database into different groups and by collecting summary information about them, the performance and accuracy of database selection can be further improved. To verify the claim, a number of experiments were conducted and their results were presented in this report.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

As the number of information sources increases rapidly on the Internet, users are beginning to experience difficulties locating the relevant information sources to meet their search requirement. These information sources could be document collections, SQL databases, or other kinds of databases. Although many web search engines e.g. Yahoo![Yah], Altavista[Alt], etc., are available on the Internet, they are only useful for discovering individual web pages instead of information sources such as document collections, SQL databases, etc.. Web search engines index all web pages found on the Internet and support keyword searches on the constructed indices. They, however, cannot be easily extended to index the content of information sources for several reasons:

- It may not be possible for information sources to provide all their content available on the Internet due to copyright or business reasons. Access to these information sources is only possible via some pre-defined query interfaces.
- To index web pages from the Internet, web search engines employ some robot agents which navigate themselves from discovered and indexed web pages to new undiscovered web pages. Since these robot agents are programmed to discover information through navigation only, they are incapable of discovering information via the query interfaces provided by information sources.
- The information sources may contain data in many different forms other than text. For example, some of them may be SQL databases; others may contain data in proprietary formats. These spectrum of data formats cannot be easily handled by the web search engines.

We call the entire process of selecting information sources to be queried, forwarding queries to the different selected sources, and merging their query results **query routing**. An

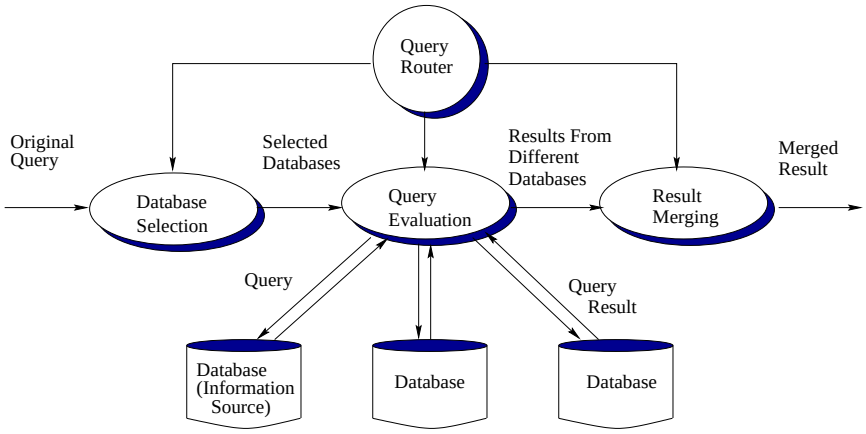


Figure 1: Query Routing Steps

intelligent agent that is designed to perform query routing can be called a *query router*. As shown in Figure 1, *database selection*, *query evaluation* and *result merging* are the three main steps involved in query routing. In the database selection step, the query router chooses the best information source(s) to evaluate a query based on some knowledge about the sources. The query is then submitted to these information sources. Some transformations may have to be performed on the queries and their results if the information sources adopt query and result formats different from that used by the global user. In the result merging step, the query router merges the results returned by different information sources. Re-computation of the rank of result records may have to be performed when ranking is required for the merged result.

Depending on the type of search requirement and the type of data sources, different variants of query routing problems can be defined. In the following, we illustrate some specific examples of query routing problems.

- *Scenario 1 (Global Digital Library System)*: In a global digital library system that is built upon multiple text collection servers on the Internet, a query routing problem may be defined by determining the most relevant text collection(s) for any user query which includes keyword criteria specified on the title, author and/or subject of the distributed text documents. Query routing, in this case, may involve searchable or non-searchable text collections, and the results from selected text collections may have to combined together in order to form a single result set for the original query.

- *Scenario 2 (Electronic Shopping)*: The Internet is predicted to be a commonplace for users to perform electronic shopping. The promise of electronic shopping depends to a large extent upon the user interface and how users interact with the various electronic commerce agents on the Internet. Typically, each retailer will provide online information about his/her products. To ensure that a buyer with a specific buying

need will be able to locate the right retailer(s) quickly, we need a query routing mechanism that can suggest fairly accurately a small number of retailers for the buyer to consider or patronize. Unlike scenario 1, the data sources to be dealt with are product information from the retail stores, and the buying need may be represented by conditions specified on the product attributes.

In solving query routing problems, a number of important issues also have to be considered:

- *Heterogeneity:* Information sources are allowed to have different data formats and query interfaces. Databases may have different attribute sets. Although they may share some common attributes, the name and domain of the common attributes may still be different. Before query routing can be performed on the heterogeneous information sources, the query interface and attribute mismatches must be dealt with. In fact, query routing techniques must be developed for different classes of information sources.

- *Autonomy:* It is neither possible nor feasible to maintain complete knowledge about the information sources. Leaving the storage issue aside, the owners of information sources usually do not wish to sacrifice their control over the information content and to reveal the operational details of their information sources.

- *Content Evolution:* The content of information sources may vary after some updates are performed. This dynamic changes of content may jeopardize the query routing knowledge constructed based on past queries or past query results.

1.1 Scope and Objectives

In recent years, query routing problems for collections of text-based databases have received a lot of attention. In contrast, there has not been much query routing research performed on other kinds of information sources.

Bibliographic databases represent an important class of information maintained by existing library systems. We envisage a future whereby a large number of bibliographic databases will be created, maintained, and made available on the Internet by content providers, publishers, and librarians. As each bibliographic database may be designed to store different types of bibliographic records, a meta-search mechanism[GCGM97] has to be provided by a query router to assist digital library users in their quest of information.

In this report, we focus on addressing the *database selection* problem for a set of bibliographic databases each containing a set of records with multiple text-based attributes.

The bibliographic database engines involved support simple boolean queries and the results they return are unranked.

We aim to develop new database selection techniques based on two classes of approaches. The first class of proposed database selection techniques exploits the use of training queries to construct the knowledge base for query routing. The second class of techniques, on the other hand, incorporate different clustering techniques to improve the performance of database selection. The performance of these two classes of proposed database selection techniques together with techniques proposed by other researchers are further compared using some simulated bibliographic database collections.

1.2 Problem Definition

There are many variants of database selection problems for a set of bibliographic databases. In general, they can be formally defined as follows:

Definition 1 *Let $D = \{db_1, db_2, \cdots, db_N\}$ be a set of bibliographic databases. Let q be a query and M be the number of databases to which query q should be forwarded to. The* **M-Database Selection Problem** *is defined as the following optimization problem.*

Compute $E \subseteq D$ such that $|E| = M$ and ($\forall F \subseteq D$ such that $|F| = M$, $Goodness(q, E) \geq Goodness(q, F)$)

Here, *Goodness* is a function on the results returned by a set of databases. Depending on the definition of *Goodness*, different variants of M-Database Selection Problems can be derived. M is a number determined by the global users or applications to constrain the number of databases for evaluating the query. The database selection problem is trivial when $M = N$.

In the rest of this report, we are concerned with maximizing the result sizes returned by the selected databases. Hence, given a set of databases E and a query q , we define our **Goodness** as follows:

Definition 2

$$Goodness(q, E) = \sum_{i \in E} s_i$$

where s_i denotes the result size returned by db_i for query q.

We have also adopted a simple keyword-based boolean query model for the bibliographic databases. For example, to retrieve all bibliographic records having title keywords "database" and subject keywords "information" and "retrieval", the following query can be formulated.

title="database" and subject=("information" and "retrieval")

At present, we have restricted our queries to contain only conjunctions of predicates on title and subject. Nevertheless, our proposed techniques can be easily extended to handle other text-based attributes.

1.3 Outline

The rest of this report is structured as follows. In Chapter 2 we provide a brief survey of the relevant work. Chapter 3 discusses our proposed database selection techniques that are based on training queries. Chapter 4 further introduces our proposed cluster-base database selection techniques. The experiments conducted for the two classes of proposed database selection techniques are reported in the respective chapters. Chapter 5 concludes the report and describes our future work.

Chapter 2

LITERATURE SURVEY

In recent years, different forms of database selection problems have been studied by several research groups, and a number of solution approaches have been proposed. As pointed out in [GP98], database selection problems can occur both in *routing* and *mediating* queries to distributed set of data sources. Query routing often operates on top of a set of text collections or collections with text attributes such that the collections share a common and simple schema. Query mediation, on the other hand, involves heterogeneous schemas exported by the underlying databases and the schemas usually complement one another in the database content. While query routing often adopts a query model which returns partial results to any given queries, query mediation requires complete query results to be returned from the participating databases.

In the following, we describe previous research efforts in database selection for query routing. These research efforts can be classified into three main categories according to the type of data sources.

2.1 Database Selection for Text Collections

Research efforts in this category deal with collections of text documents. Usually, the *vector space* retrieval model is adopted for querying the text collections. A query supported by such model consists of a set of keywords, and the relevance of a text document is determined by the frequency of keywords appearing in the document and their discriminatory power.

In the gGlOSS project[GGM95], the document frequency for each word in each text collection is extracted and included in the knowledge base for database selection. Using the document frequencies, the relevance of each text collection can be estimated for a given user query.

In Callan's work, the CORI (*Collection Retrieval Inference Network*) project[CLC95], the $TF \times IDF$ document ranking method has been extended to rank a set of text collections where TF denotes *term frequency* and IDF denotes *inverse document frequency*. In this method, the $TF \times IDF$ document scoring formula is modified by replacing TF and IDF by DF and ICF (*inverse collection frequency*) respectively. A CORI network is later constructed based on the relationship between collections and their terms, and the relationship between a given query and its term. Each collection is scored using the CORI network and is determined by the combined belief or probability of all query terms. It is assumed that all terms involved in the query are of equal importance.

Based on the document frequency knowledge, Yuwono and Lee proposed a unique database ranking formula based on Cue-Validity Variance (CVV). The proposed database ranking formula essentially incorporates the discriminatory power of keywords across collections. It was shown that the CVV-based database selection technique out-performed the database selection techniques in gGlOSS and CORI.

2.2 Database Selection for Collections with Multiple Attributes

In the GlOSS (*Glossary of Servers Server*) project[GGMT94, TGL+97], a database selection technique for collections containing multiple text attributes has been proposed. The queries for such collections consist of keyword predicates on the different attributes such as *author*, *title*, etc. Given a collection and an attribute-term pair, the number of records having the attribute values containing the term is known as the *frequency of the attribute-term pair*. This frequency information has been further used to estimate the rank of each database. The main assumption behind GlOSS is that terms appearing in any specific attribute of records of a collection follow independent and uniform distributions. The discriminatory power of each term is not considered in this work. Real user queries and a set of six databases(INSPEC, COMPENDEX, ABI, GEOREF, ERIC and PSYCINFO databases) have been used to evaluate the performance of GlOSS.

Li and Danzig [LD97] proposed a database selection technique based on calculating similarities between databases and a boolean query. His research requires that both user queries and database descriptions to be written in boolean expressions. The similarity measure will be further applied to these boolean expressions and used to rank databases. However, boolean expressions may not precisely describe databases' content when these databases contain huge number of records with multiple text attributes. This proposed database selection technique has been experimented on two database collections(CISI and USC Homer).

In this report, we will present new database selection techniques that exploit the discrimintory power of attribute-term pair. As part of our experiments, we have also developed an approach to generate simulated databases such that their content skewness can be controlled.

2.3 Database Selection for Collections Accessible Through Query Interface Only

So far, the database selection techniques given in Sections 2.1 and 2.2 assume that the document frequency information for each text collection is available for query routing. This is possible either by having full access to the text collections or by mandating each text collection to provide the necessary information voluntarily. Nevertheless, in reality, not all text collections may be able to cooperate fully on providing their local information. Hence, one may have to investigate database selection techniques for collections accesible through query interface only.

Voorhees[TVGJL95] proposed two query routing techniques in the domain of unstructured document collections. In the two techniques, known as *multiple relevant document distribution* (MRDD) and *query clustering* (QC), document collections are ranked based on their responses to the training queries most similar to the query to be routed. These methods are cost efficient in terms of resource utilization and implementation effort. However, it is not clear how training queries that sufficiently capture the content of a document can be generated. Furthermore, the two techniques only deal with document collections. In our research, we will propose database selection techniques for bibliographic databases such that the techniques require training query information only.

Chapter 3

DATABASE SELECTION TECHNIQUES BASED ON TRAINING QUERIES

In this chapter, three database selection techniques based on training queries will be presented. They are known as the:

- Database selection technique based on training queries and their result sizes(TQS);
- Database selection technique based on training query result summary (TQRS); and
- Database selection technique based on training query result summary using GlOSS (TQRG)

To evaluate and compare their performance, a experimental framework has been established. Moreover, we will also introduce two non-training query based database selection techniques using GlOSS and cue validity variance (CVV) as the baseline techniques for performance comparison.

In Sections 3.1 and 3.2, the two non-training query based database selection techniques are given. The TQS, TQRS and TQRG techniques are later described in Section 3.3, 3.4 and 3.5 respectively. The performance experiments and the experiment results are given in Section 3.6.

3.1 Database Selection based on GlOSS(GlOSS)

In the database selection techniques based on GlOSS (*Glossary of Servers Server*) [GGMT94], the estimated result size of a boolean query q returned from a database db_i, $\widehat{Size}_{(db_i,q)}$, is computed from *tuple frequencies* summarized from the entire database content. Similar to document frequency, tuple frequency (denoted by $TF_{i,j,k}$) is defined as the number of tuples(records) in database i containing term j in the bibliographic attribute A_k.

3.1.1 Database Ranking

The GlOSS technique defines the **goodness score** $G_{db_i,q}$ for a given database db_i and a given boolean query q as follows:

$$G_{db_i,q} = \widehat{Size}_{(db_i,q)} = N_i \prod_{k=1}^{|A|} \prod_{j=1}^{|A_k|} \frac{TF_{i,j,k}}{N_i} \tag{1}$$

where $|A_k|$ denotes the number of terms in the domain of attribute A_k, and N_i denotes the number of tuples in database db_i.

3.1.2 Analysis of GlOSS Technique

This technique extracts statistical information (i.e, tuple frequencies) from each bibliographic database and also store the total number of tuples in each bibliographic database in the knowledge base for database selection. Although the technique acquires a rather complete knowledge about the bibliographic database, it violates some degree of local autonomy. This may not desirable to some bibliographic database systems.

3.2 Database Selection based on Database Summary Information(DS)

The DS technique is developed based on the CVV server ranking method proposed in [YL97]. It exploits the discriminatory powers of terms to improve the accuracy of database selection. Like GlOSS, it also relies on the tuple frequencies and the number of tuples in each bibliographic database.

3.2.1 Database Ranking

The *Cue-Validity Variance* (CVV) ranking technique was proposed by Lee to rank a set of document databases[YL97]. The DS technique essentially extends CVV ranking technique to rank databases containing multiple text attributes.

Given a set of databases D, the DS technique assigns a goodness score $G_{db_i,q}$ to database $db_i \in D$ with respect to query q as follows:

$$G_{db_i,q} = \prod_{k=1}^{|A|} \sum_{j=1}^{|A_k|} CVV_{j,k} \cdot TF_{i,j,k} \tag{2}$$

where $CVV_{j,k}$ is the variance of $CV_{i,j,k}$'s, the *Cue Validity* of term j, across all databases for attribute A_k in A(={title, subject}). $|A_k|$ denotes the number of terms for attribute A_k.

$$CV_{i,j,k} = \frac{\frac{TF_{i,j,k}}{N_i}}{\frac{TF_{i,j,k}}{N_i} + \frac{\sum_{l \neq i}^{|D|} TF_{l,j,k}}{\sum_{l \neq i}^{|D|} N_l}} \tag{3}$$

where N_i is the number of tuples in database db_i, and $|D|$ is the number of databases in the system. The population variance $CVV_{j,k}$ of $CV_{i,j,k}$ measures the skewness of the distribution of term j of attribute A_k for distinguishing one database from another for particular attribute A_k. The larger is the variance, the more discriminatory is the attribute term. $CVV_{j,k}$ is computed as follows.

$$CVV_{j,k} = \frac{\sum_{i=1}^{|D|} (CV_{i,j,k} - \overline{CV_{j,k}})^2}{|D|} \tag{4}$$

where $\overline{CV_{j,k}}$ is the population mean of $CV_{i,j,k}$ over all databases for attribute A_k, and is defined as follows.

$$\overline{CV_{j,k}} = \frac{\sum_{i=1}^{|D|} CV_{i,j,k}}{|D|} \tag{5}$$

3.2.2 Analysis of DS Technique

To deploy DS technique, one has to extract statistical information (i.e. tuple frequencies) from each bibliographic database and store them as part of the knowledge base for database selection. Like GlOSS technique, DS also requires a rather complete knowledge about the bibliographic database. This may not desirable to some bibliographic database systems.

3.3 Database Selection based on Training Queries and Their Result Sizes (TQS)

Unlike GlOSS and DS techniques, this technique attempts to discover the content of each bibliographic database using a set of training queries, and estimate the result size of a new query using that of similar training queries. Hence, we construct for each bibliographic database a knowledge base consisting of the set of training queries and their result sizes. Formally, the knowledge base $KB(db_i)$ for database db_i can be represented as follows:

$$KB(db_i) = \{(tq_1, s_{i,1}), (tq_2, s_{i,2}), \cdots, (tq_p, s_{i,p})\}$$

where each $tq_j (1 \leq j \leq p)$ is a training query and $s_{i,j}$ denotes the result size of tq_j returned by database db_i.

3.3.1 Knowledge Base Construction

To construct the knowledge base, a set of training queries has to be generated. There are essentially two possible approaches to do so. One can either collect past queries as training queries, or create synthetic training queries. In this research, we have chosen the latter due to three reasons: it is time consuming to collect queries; the collected queries may not have a good coverage of the bibliographic database content; and database owners are often apprehensive towards modification to existing query interface so that the past queries can be logged. The training queries we generated for the knowledge base satisfy a number of criteria:

- Each training query is a boolean query consisting of a conjunction of keyword predicates on the *title* and/or *subject* attributes. Other bibliographic attributes such as *author* and *publisher* have been excluded because these attributes were not considered in the construction of our experimental collections as described in the later part of this report. However, for real collections such as those used by the NCSTRL (Networked Computer Science Technical Reports Library) [NCS], author and other attributes may provide important information for database selection.

- Each training query must return sizable result at least for a bibliographic database before it can be included in the knowledge base. The minimum result size is required since queries returning very small results do not capture a database content well. The minimum result size that must be satisfied by training queries is specified by the parameter L_{tr}. In our experiment, we empirically choose L_{tr} to be 4 since the TQS technique performs well with this number[1].

[1] The same constraint also works well for the TQRS technique.

The synthetic training queries are generated as follows.

- *Step 1:* Randomly select a record from the combined set of bibliographic records collected from all experimental databases.
- *Step 2:* Extract title and subject values from the record. (Note that it is possible for a bibliographic record to have multiple subjects.)
- *Step 3:* Randomly decide whether to use title, subject or both in a new training query.
- *Step 4:* For each attribute (title or subject) to be included in the training query, construct a predicate on it by randomly selecting one to four distinct terms from the corresponding extracted attribute value[2]. No stop words are used in this step.

For example, a randomly selected bibliographic record is entitled "Algorithms and Data Structures in C++" and has multiple subjects "C++ (Computer programming language)", "Computer algorithms", and "Data structures (Computer science)". A training query consisting of the following predicates may be generated:

title = ("Algorithms" and "Structures" and "C++")
subject = ("Computer" and "language")

At present, we have not thoroughly investigated the storage space constraint imposed on knowledge bases for the bibliographic databases. The storage space constraint essentially creates two sub-problems: *allocation of storage space to each knowledge base*; and *choice of training queries included in the knowledge base*. A simple space allocation strategy assigns storage space to knowledge bases according to their bibliographic database sizes. Since training queries with large result sizes carry more significant knowledge about the bibliographic databases, they should be given preference among training queries to be included in the knowledge base.

3.3.2 Database Ranking

To rank bibliographic databases for a query using the TQS technique, we estimate the result size returned by every bibliographic database using training queries from the corresponding knowledge base. Before we formally present the proposed database ranking technique, the following term are first defined.

- **Matching selection predicates**

[2]Since a bibliographic record may have multiple subject values, all terms are selected from one of the subject values.

Definition 3 *Two selection predicates $p_1 \equiv (A_1 = val_1)$ and $p_2 \equiv (A_2 = val_2)$ are said to match if $A_1 \equiv A_2$, where A_1 and A_2 are attribute names, and each val_i ($i = 1$ or 2) represents a conjunction of terms.*

- **Predicate similarity measure**

  Similarity between two matching predicates is defined as the similarity between the values used in the predicates, which is measured by the cosine distance[FBY92, Sal88] defined as follows:

  Definition 4 *Let $p_1 \equiv (A = val_1)$ and $p_2 \equiv (A = val_2)$ be two matching predicates. The similarity measure of p_1 and p_2, denoted by $simp(p_1, p_2)$, is defined below.*

$$simp(p_1, p_2) = \frac{|val_1 \cap val_2|}{\sqrt{|val_1|} \cdot \sqrt{|val_2|}} \tag{6}$$

  where $|val_1|$ and $|val_2|$ refer to the numbers of distinct terms in p_1 and p_2 respectively, $|val_1 \cap val_2|$ refers to the number of distinct terms val_1 and val_2 have in common.

  The definition of cosine distance implies that predicate similarity measure is in the range of [0,1]. For example, let predicates p_1 and p_2 be *title = ("database" and "selection")* and *title = ("database" and "design")* respectively. The similarity between the two predicates is $\frac{1}{\sqrt{2} \cdot \sqrt{2}}$.

- **Query similarity measure**

  Definition 5 *Let q_1 and q_2 be two queries, $P(q)$ be all selection predicates in a query q, and $MP(q_1, q_2)$ be the set of matching predicate pairs (i.e. $MP(q_1, q_2) = \{(p_1, p_2) | p_1 \in P(q_1), p_2 \in P(q_2)$, p_1 and p_2 are matching predicates$\}$). The similarity measure between q_1 and q_2, denoted by $simq(q_1, q_2)$, is defined as:*

$$simq(q_1, q_2) = \frac{2 \cdot \sum_{i=1}^{k} simp(p_1, p_2)}{|P(q_1)| + |P(q_2)|} \tag{7}$$

  where $(p_1, p_2) \in MP(q_1, q_2)$, $k = |MP(q_1, q_2)|$.

  In above definition, $simp(q_1, q_2)$, which is the normalized sum of predicate similarity of all matching predicates in q_1 and q_2 , will yield a value in the range of [0,1] because

$$k = |MP(q_1, q_2)| \leq min(|P(q_1)|, |P(q_2)|)$$

  .

Let Q denotes a set of training queries. Given a query q, the TQS technique defines the goodness score for a database db_i as follows:

$$G_{db_i,q} = \widehat{Size}_{(db_i,q)} = \frac{\sum_{j=1}^{p} simq(q, tq_j) \cdot s_{i,j}}{\sum_{j=1}^{p} simq(q, tq_j)} \tag{8}$$

where $tq_1, \cdots, tq_p \in Q$, $s_{i,j}$ denotes the stored result size of tq_j return by database db_i.

Note that all databases use the same set of training queries.

3.3.3 Analysis of TQS Technique

In the TQS technique, query similarity measure is computed based on the assumption that all terms in the matching predicates are of equal importance, i.e. they have the same discriminatory power. This assumption do not usually hold for a bibliographic database. Furthermore, the discriminatory power of terms may be different across bibliographic databases. TQS technique relies on training queries and their result sizes. Instead of using bibliographic database records to create synthetic training queries, one can easily modify the technique to use some dictionaries for title and subject attributes. This allows TQS technique to operate without violating the autonomy of existing bibliographic database systems.

3.4 Database Selection based on Training Query Result Summary(TQRS)

TQRS technique combines both TQS and DS techniques. Instead of directly collecting database summary information from a set of bibliographic databases, TQRS technique uses a set of training queries to sample the content of bibliographic databases, and to build database summary information using the training query results.

3.4.1 Database Ranking

One first obtains a set of synthetic training queries using the query generation strategy of TQS technique. The combined result of these training queries thus represents a sample of the database content from which tuple frequencies and cue-validity variances can be computed. Using a goodness definition similar to (2), the bibliographic databases are ranked. Formally, the goodness score $G_{db_i,q}$ for database $db_i \in D$ with respect to query q is defined by the TQRS technique as follows:

$$G_{db_i,q} = \prod_{k=1}^{|A|} \sum_{j=1}^{|A_k|} CVV'_{j,k} \cdot TF'_{i,j,k} \tag{9}$$

where $TF'_{i,j,k}$ denotes the tuple frequency of term j for attribute A_k computed from the combined training query result for db_i; and

$$CV'_{i,j,k} = \frac{\frac{TF'_{i,j,k}}{N'_i}}{\frac{TF'_{i,j,k}}{N'_i} + \frac{\sum_{l \neq i}^{|D|} TF'_{l,j,k}}{\sum_{l \neq i}^{|D|} N'_l}} \tag{10}$$

where N'_i is the number of tuples in the combined training query result for db_i.

$$CVV'_{j,k} = \frac{\sum_{i=1}^{|D|}(CV'_{i,j,k} - \overline{CV'_{j,k}})^2}{|D|} \tag{11}$$

$$\overline{CV'_{j,k}} = \frac{\sum_{i=1}^{|D|} CV'_{i,j,k}}{|D|} \tag{12}$$

3.4.2 Analysis of TQRS Technique

The ranking procedure of TQRS technique resembles that of DS technique. However, it computes the relevant statistical information from a smaller set of database records sampled by collecting training query results. Like the case of TQS technique, the training queries generation should provide a reasonable uniform coverage of the database content. When the training query results are collected from a bibliographic database, the individual result tuples should be identified. Note that duplicate result tuples should not be permitted in the combined training query result. This can be achieved by examining unique ids (e.g. ISBN number, system id) assigned to bibliographic records.

3.5 Database Selection based on Training Query Result Summary Using GlOSS(TQRG)

This technique is derived from the database selection techniques based on TQS and GlOSS. Instead of directly collecting statistical information from a set of bibliographic databases, like TQRS, TQRG technique uses a set of training queries to sample the content of bibliographic databases, and builds the statistical information from the training query results.

3.5.1 Database Ranking

Using the combined result of a set of synthetic training queries generated by the query generation strategy of TQS technique, one can build a sample of the database content from which the tuple frequencies and the total number of tuples in each database will be computed. Using a goodness score definition similar to (1), the TQRG technique assigns the goodness score $G_{db_i,q}$ to database db_i with respect to the given query q as follows:

$$G_{db_i,q} = \widehat{Size}_{(db_i,q)} = N'_i \prod_{k=1}^{|A|} \prod_{j=1}^{|A_k|} \frac{TF'_{i,j,k}}{N'_i} \tag{13}$$

where $|A_k|$ denotes the number of terms for attribute A_k, $TF'_{i,j,k}$ denotes the tuple frequency of term j for attribute A_k computed from the combined training query result for database db_i, N'_i is the number of tuples in the combined training query result for database db_i.

3.5.2 Analysis of TQRG technique

Although the ranking procedure of TQRG technique resembles that of GlOSS technique, it computes the relevant statistical information from a smaller set of database records sampled by collecting training query results. Unlike TQRS, this technique does not need to compute the CVV values of terms.

3.6 Experiments

To evaluate the proposed three database selection techniques and the two baseline database selection techniques, we have conducted a set of experiments to evaluate their performance. The experiments have been conducted to answer a few questions about the five techniques:

- How do the techniques perform for different query requirements (e.g. by varying the number of databases to be selected M)?
- In the case of TQS, TQRS and TQRG, how does the choice of training queries affect the performance?
- How do the techniques perform for bibliographic databases with different skewness in their content?
- How do the techniques perform for different sets of test queries which have different result sizes from bibliographic databases?

In the rest of this section, we describe the experiment setup, and the performance measures used. The experiment findings are presented and analyzed.

3.6.1 Experiment Framework

To set up the bibliographic database collection for our experiments, we down-loaded all bibliographic records from NTU[3] library database. NTU library database contains 217,928 bibliographic records. The records are classified according to the Library of Congress(LC)

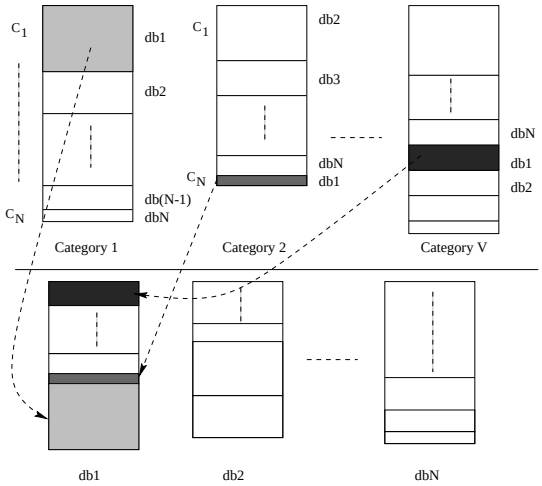


Figure 2: Organize the catalogue records from categories to databases

classification scheme. For example, the call number QA76.9.D3.AI49 indicates that the bibliographic record belongs to the *mathematics science* category.

For each of our experiments, a set of $N = 10$ bibliographic databases has been constructed using the down-loaded bibliographic records based on the following strategy:

- All bibliographic records are grouped according to their LC categories. Assume that there are V such virtual categories and we want to assign their records to N databases such that each database contains records from all categories, and at the same time contains distinct makeup of records from different categories. In this way, the databases in our collection always demonstrate different degrees of relevance for the same query.

- We divide each category into N groups, with records assigned to the groups according to the following pre-defined ratio(the sizes of these groups are determined by the Zipf-like distribution[HLY93, Gol84, VF95]):

  $$|C_1| : |C_2| : \ldots : |C_N|$$

  where:

  $$|C_i| = \frac{|C|}{i^{Z_d} \sum_{j=1}^{N} \frac{1}{j^{Z_d}}} \tag{14}$$

  $|C|$ is the size of the category, Z_d is **Database Skew**. (When $Z_d > 0$, $|C_i|$ has a Zipf-like distribution, and when $Z_d = 0$, it is a uniform distribution.)

- The groups are assigned to N databases in a round-robin manner.

[3]The web page of Nanyang Technological University library is available at:(http://web.ntu.ac.sg/library/)

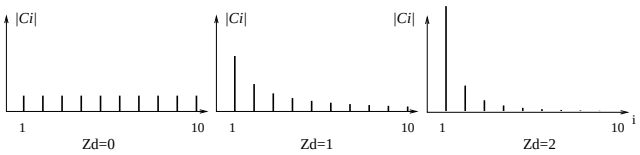


Figure 3: Category distribution given different database skew value (when N=10)

The assignment of bibliographic records to different databases in our experiment is illustrated by Figure 2.

By varying the Z_d value, we can evaluate the performance of database selection techniques in database collections with different skewness. When $Z_d = 0$, each category is evenly distributed to the N databases. It should be noted that the larger Z_d is, the more skew is each category being grouped[HLY93] (see Figure 3). In particular, $Z_d = 1$ was selected as a normal database skew level so that we can evaluate the performance of our techniques when a static database skew is required. On the other hand, the different degrees of database skew, $Z_d = 0$, 0.5, 1, 1.5, and 2 were used in our experiments to evaluate the performance of our techniques for database collection with different database skews.

In our experiments, we generated 8000 training queries and 2000 test queries. Each query is generated using the training query generation procedure in TQS technique. Note that the training query set and test query set are distinct.

3.6.2 Performance Measurement

In our experiments, two performance measures have been adopted. The first performance measure (denoted by P) derives the accuracy of a database selection technique by computing the ratio between the combined result size returned by the database selection technique and that returned by the ideal choice of databases.

Definition 6 *Given K test queries $\{q_1, q_2, \cdots, q_K\}$, the* **performance** *$P$ is computed as follows:*

$$P = \frac{1}{K} \sum_{j=1}^{K} P_j \tag{15}$$

where $P_j (1 \leq j \leq K)$ represents the performance contributed by test query q_j.

$$P_j = \frac{\sum_{db_i \in G} s_{i,j}}{\sum_{db_i \in B} s_{i,j}} \tag{16}$$

G represents the set of databases selected by a proposed database selection technique. The ideal database selection is B. $s_{i,j}$ denotes the actual result size of test query q_j returned by database db_i.

Clearly $0 \leq P \leq 1$. When $M = N$, $G = B$ and $P = 1$[4].

The second performance measure (known as mean-square rank error denoted by P') determines the difference between the predicted ranks and the actual ranks of databases[CLC95].

Definition 7 *The* **mean-square rank error** *metric for a single test query q_j is defined as:*

$$P'_j = \frac{1}{|G|} \cdot \sum_{db_i \in G} (O_{i,j} - R_{i,j})^2 \tag{17}$$

where:
$O_{i,j}$ = actual rank for database db_i based on the actual result size returned by database db_i for test query q_j (the database with the largest number of results is ranked 1, the database with second largest number of results is ranked 2, and so on);
$R_{i,j}$ = the rank of database db_i determined by our techniques for test query q_j.

P' is derived from P'_j in the same way as P is derived(see (15)).

3.6.3 Parameter Setting

The experiments are conducted by varying the following four parameters:

- Z_d - the degree of database skew according to Zipf-like function. (Z_d = 0, 0.5, 1, 1.5, 2 were selected)
- T(for TQS, TQRS and TQRG techniques only) - number of training queries used to generate the knowledge base(for TQS technique) or queries result summary information(for TQRS and TQRG techniques). We use five different T values, namely, 100, 500, 2000, 5000, and 8000; (e.g. $T = 500$ means that the first 500 training queries are selected [5])
- M- the number of databases to be selected(M = 1, 2, ..., 10)
- L - the minimum result size for the test queries. The result size of a test query is defined by the total number of records returned by all databases. Given a L, K test queries were selected from 2000 generated test queries and used to conduct the experiment.

T	100	500	2000	5000	8000
P_{TQS}	.739	.767	.798	.808	.812
P_{TQRS}	.798	.840	.862	.865	.867
P_{TQRG}	.786	.843	.875	.881	.883
P_{GlOSS}	0.888				
P_{DS}	0.870				
P_{random}	0.574				
P'_{TQS}	12.7	11.3	10.1	9.60	9.32
P'_{TQRS}	9.14	7.23	6.50	6.40	6.32
P'_{TQRG}	9.39	7.23	6.36	6.22	6.15
P'_{GlOSS}	6.14				
P'_{DS}	6.23				
P'_{random}	16.3				

Table 1: Performance measures of the five techniques as a function of the number of training queries used T(M=5, $Z_d = 1$, P_X denotes performance P of X technique)

3.6.4 Experiments Findings

Figures 4 to 9 show the performance of the five different database selection techniques against the number of databases to be selected (M) when $L = 2, Z_d = 1$. The performance of random database selection, which randomly selects M databases out of N databases, serves as a bottom baseline for the performance of all database selection techniques. Both the GlOSS and DS techniques have been used as the top baselines for the performance of TQRG and TQRS, respectively[6]. For TQRS and TQRG techniques, training query results were used to generate result summary information. The total numbers of tuples used by TQRS(TQRG) were shown in Figures 6 and 7 (Figures 8 and 9).

- As shown in these figures, GlOSS and DS techniques yield the best performance.
- Our proposed techniques always outperform the random database selection.
- For TQS, TQRS and TQRG techniques, as the number of training queries stored in the knowledge base(for TQS) or used for database sampling (for TQRS and TQRG) increases, their performance also improve. Especially for TQS technique, the performance using 8000 training queries improves more than 50% over that using 100 training queries when M=1 (from 0.43 to 0.64). Table 1 shows the performance measures of the five techniques and the random database selection techniques when $M = 5$. GlOSS and DS techniques clearly give the upper bounds for TQRG and TQRS techniques, respectively. When the number of training queries is large enough, TQRG (or TQRS) technique uses all tuples in databases to generate result summary information. Hence it behaves like the GlOSS (or DS) technique.

[4] $M = |G| = |B|$. M is defined in *Introduction* section.

[5] The set of training queries selected by $T = 100$ is a subset of training queries selected by $T = 500$, which is a subset of training queries selected by $T = 2000$, and so on.

[6] We will further denote GlOSS and TQRG techniques by GlOSS/TQRG, DS and TQRS techniques by DS/TQRS, respectively.

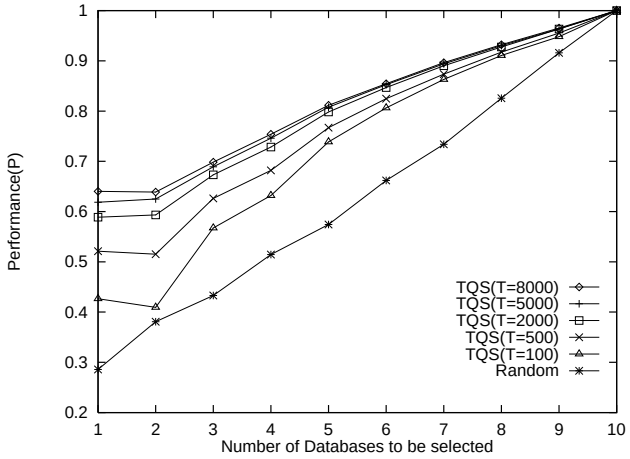


Figure 4: Performance P of TQS technique with 1224 test queries as a function of the number of database to be selected M (L=2 , $Z_d = 1$)

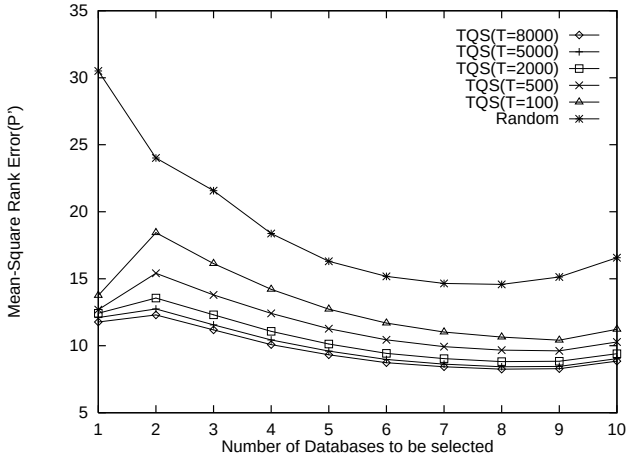


Figure 5: Mean-square rank error P' of TQS technique with 1224 test queries as a function of the number of database selected M (L=2, $Z_d = 1$)

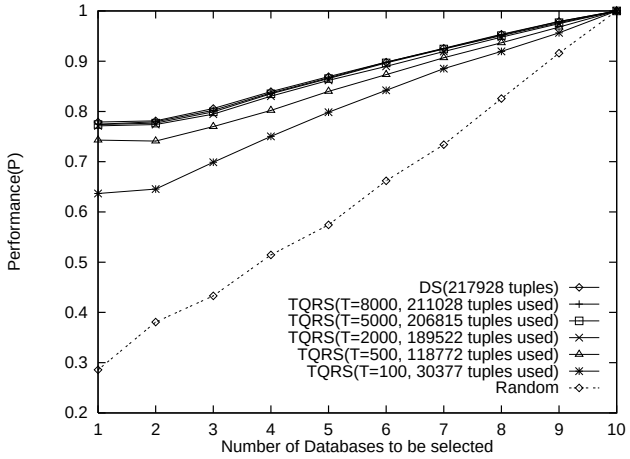


Figure 6: Performance P of DS and TQRS techniques with 1224 test queries as a function of the number of database selected M (L=2, $Z_d = 1$)

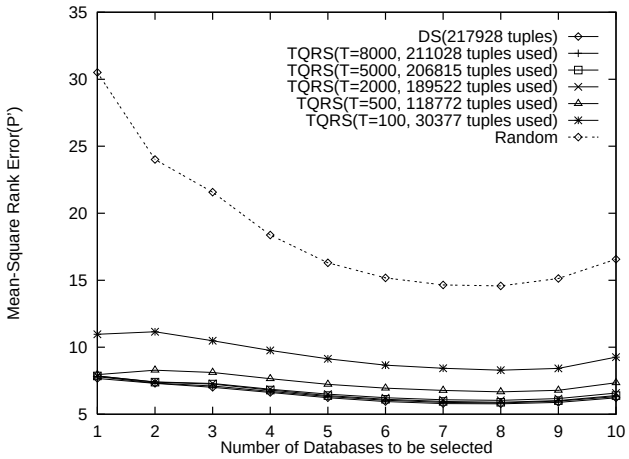


Figure 7: Mean-square rank error P' of DS and TQRS techniques with 1224 test queries as a function of the number of database selected M (L=2, $Z_d = 1$)

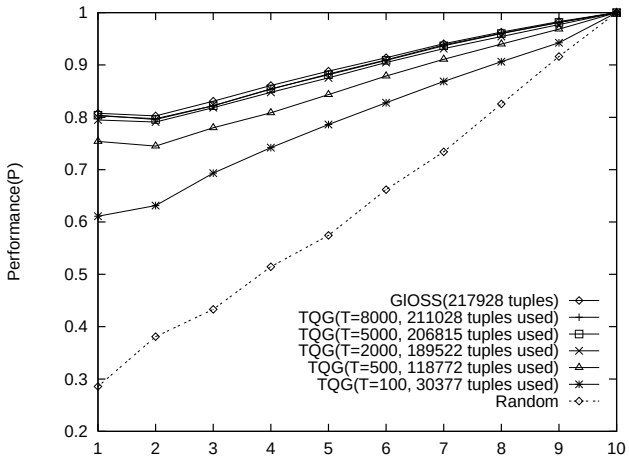


Figure 8: Performance P of GlOSS and TQRG techniques with 1224 test queries as a function of the number of database selected M (L=2, $Z_d = 1$)

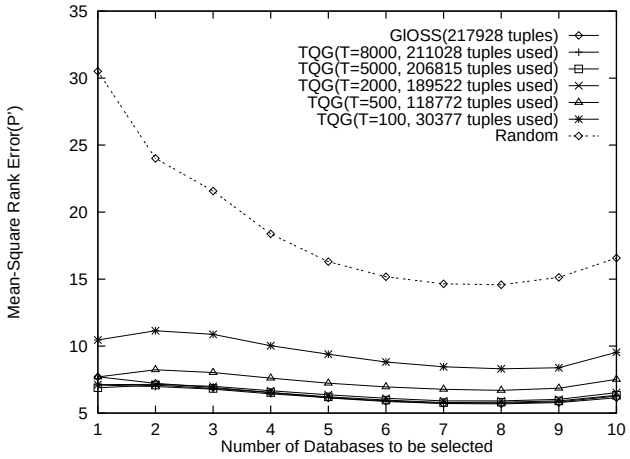


Figure 9: Mean-square rank error P' of GlOSS and TQRG techniques with 1224 test queries as a function of the number of database selected M (L=2, $Z_d = 1$)

L	2	10	20	30	40	50	60
P_{GlOSS}	.888	.952	.971	.975	.979	.981	.983
P_{TQRG}	.883	.948	.968	.974	.978	.980	.982
P_{DS}	.869	.945	.966	.972	.977	.979	.981
P_{TQRS}	.866	.943	.965	.970	.977	.979	.981
P_{TQS}	.812	.884	.903	.910	.917	.921	.923
P_{random}	.585	.646	.666	.666	.668	.677	.663
K	1224	695	560	504	453	420	383

Table 2: Performance P and the number of test queries used K as a function of threshold L(M=5, T=8000(for TQS, TQRS and TQRG only), $Z_d = 1$)

- TQRG and TQRS techniques always outperform TQS technique. In particular, TQRG(or TQRS) technique using only 100 training queries (only 14% of all tuples in databases were used) performs as well as TQS technique using 8000 training queries. Our experiments show that GlOSS/TQRG and DS/TQRS techniques are able to capture the content feature of database better than the TQS technique which relies mainly on query similarities.
- GlOSS/TQRG techniques slightly outperform DS/TQRS techniques.

We also discovered that when test queries with large result sizes are used in the experiment, the performance of all proposed techniques become much better as shown in Table 2. This observation is fair since it is usually more difficult to decide the relevant databases when the result of a query is very small.

To investigate the performance of our techniques in different types of database collections, we conducted experiments on collections with different database skew values. Since every experiment has to be re-conducted for each database skew, the experiment is extremely time-consuming. We have experimented with five different database skew values.

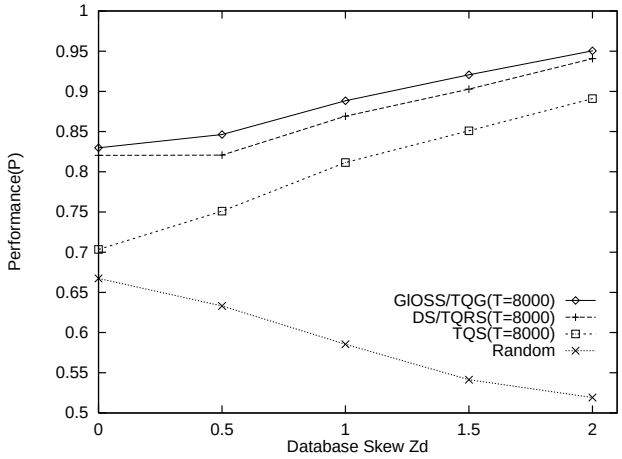


Figure 10: Performance P of different techniques as a function of database skew value Z_d (L=2, M=5, T=8000(for TQS, TQRS and TQRG only))

Nevertheless, the results of our experiments are encouraging. As shown in Figure 10 and 11, our techniques consistently outperformed random database selection significantly for all database skews. Note that the performance for TQRG with T=8000(or TQRS with T=8000) technique is so similar to the GlOSS(or DS) technique that we simply use the performance of GlOSS(or DS) as the representative. As shown in Figure 10, when databases are uniformly distributed(i.e. database skew $Z_d = 0$), GlOSS/TQRG and DS/TQRS techniques outperform random database selection by 20% while TQS technique outperforms the latter by 5%. As the database skew increases, there are significant improvement in the performance of our proposed techniques. It means that databases with large result size will be easier to be selected by our proposed techniques when the database skew is larger. On the other hand, the performance of random database selection degrades dramatically. For the database collection generated with $Z_d = 2$, GlOSS/TQRG($T = 8000$) techniques yield the best performance(about 0.95). Figure 11 shows the mean-square rank error performance P' of GlOSS/TQRG($T = 8000$), DS/TQRS ($T = 8000$) and TQS techniques. The figure reveals that the relative performance of the proposed techniques using mean-square rank error is similar to that using performance P.

The storage requirement for each technique was also investigated in our experiments. The storage size for the whole database collection is 30MB. The storage requirement for GlOSS/DS, TQRG/TQRS($T = 8000$), TQRG/TQRS($T = 100$), and TQS($T = 8000$) are 5MB, 4.8MB, 1.2MB and 570KB respectively (when $Z_d = 1$). In other words, TQS technique only need 1.9% storage size of all databases. Note that DS and TQRS store terms, tuple frequency and relevant term CVV values for each database[7]. GlOSS and TQRG store terms, tuple frequency for each databases where TQS stores only training queries and their result

[7]Actually, DS/TQRS need more storage size than GlOSS/TQRG because of the term CVV values' storage. However, it is trivial when there is a large number of databases.

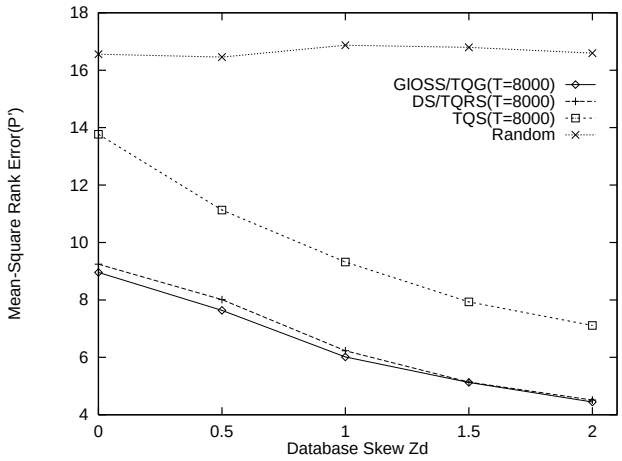


Figure 11: Mean-square rank error P' of different techniques as a function of database skew value Z_d (L=2, M=5, T=8000(for TQS, TQRS and TQRG only))

sizes. However, TQRG and TQRS allow different storage requirement when using different number of training queries. We believe that our techniques are more suitable to sets of large databases since the storage requirement of the techniques only increases marginally as the database sizes increase

Finally, we comment on the computational overhead incurred for the three proposed database selection techniques. TQRG and GlOSS require little computation compared to TQRS/DS where TQRS and DS require little computation compared to TQS. For TQS, additional computations are required to evaluate the similarity between a given query and the training queries stored in the knowledge base.

Chapter 4

CLUSTER-BASED DATABASE SELECTION TECHNIQUES

The performance of database selection largely depends on how the database content can be accurately summarized and how the summarized knowledge about databases can be further used to predict the goodness of each database with respect to a given query. In this project, we investigate the use of clustering to improve the accuracy of database selection. Several cluster-based database selection techniques have been proposed to route bibliographic queries.

Unlike other non clustered-based approaches, cluster-based database selection techniques involve clustering of database tuples before the content of each database is summarized. In this research, we have adopted three clustering techniques known as Single Pass Clustering(SPC^C), Reallocation Clustering(RC^C), and Constrained Clustering(CC^C). In Chapter 3, both the estimated result size (ERS) and estimated goodness score (EGS) have been used to rank databases. The two ranking formulas have been further adapted to the cluster-based database selection techniques. To evaluate the performance of cluster-based database selection, a number of experiments have been conducted using the experiment framework given in Section 3.6.

The overall structure of this chapter is as follows. In Section 4.1, we give an overall description of the cluster-based database selection techniques. Section 4.2 describes three database clustering techniques. Following that, two cluster-based database ranking formulas known as ERS and EGS are given in Section 4.3. The performance evaluation experiments of database selection techniques built upon combination of database clustering techniques and database ranking formulas are reported in Section 4.4.

4.1 Overview of Cluster-based Database Selection

Clustering refers to the grouping of database records based on the degrees of similarity between the records. Clustering has been used in many fields, such as information retrieval(IR)[Sal88, FBY92, Mea92], data mining, data reduction[BDF+97], etc.. In order to route queries to a set of databases each with multiple text attributes, the content of each databases has to be summarized properly. Nevertheless, as the databases contain wide range of information, direct summarization of their content may result in inaccurate summary knowledge. In such cases, clustering may be applied to discover the hidden grouping of database records. By summarizing the content of different groups of database records, we believe that the accuracy of summary knowledge can be improved.

We therefore proposed a number of cluster-based database selection techniques and apply them to the query routing problem over a set of bibliographic databases. A number of issues have to be addressed when clustering techniques are applied to the database selection problem:

- What is a cluster? How is a cluster represented?
- How are the similarity between two bibliographic records, and similarity between a bibliographic record and a cluster defined?
- What are the clustering algorithms?
- How many clusters should be generated for each bibliographic database? How does the number of clusters affect the database selection performance?

The overall steps of all cluster-based database selection techniques are depicted in Figure 12. During knowledge construction, bibliographic records from each database involved are clustered and the content of each cluster is summarized. Database ranking is therefore performed for a given query based on the summary information of the clusters. The query will be matched against the clusters during database ranking. When a query matches well with a cluster, it is likely that many records in that cluster will be relevant to the query. In this case, the rank of each database will be determined by how well its clusters match the given query, and the cluster sizes.

4.2 Clustering Techniques for Bibliographic Databases

Clustering of text documents is a well researched problem in information retrieval[CKPT92, SHS96, KS97, LD97]. Nevertheless, to our best knowledge, there has not been much previous work on clustering bibliographic databases that consist of multiple text attributes. In our research, we therefore adapted some text clustering techniques to cluster bibliographic databases.

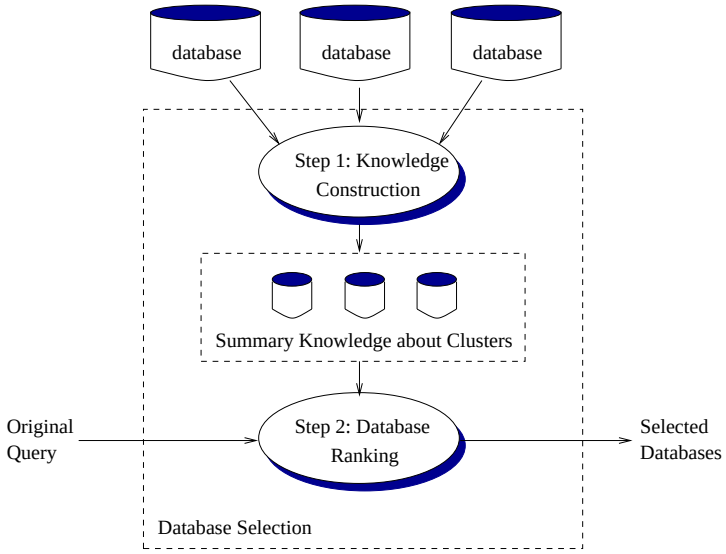


Figure 12: Cluster-based database selection steps

4.2.1 Similarity Measure Between a Bibliographic Record and a Cluster

Clustering can only be performed on the bibliographic databases when the similarity between a bibliographic record and a cluster can be determined and quantified. In this section, the similarity measures used in our proposed clustering techniques are defined.

Several attributes can be found in a bibliographic database, e.g. *title*, *subject*, etc.. An attribute value can be described by an attribute descriptor (to be defined later) which is essentially a text vector. A database record can be represented by a set of attribute descriptors, one for each attribute. Similarly, a cluster can also be represented by a set of attribute descriptors. Hence, the similarity measure between a bibliographic record and a cluster has to be defined based on the attribute descriptors information describing the record and that describing a cluster. In this report, we only deal with text attributes and assume that all databases contain the same set of attributes[1].

To support our proposed techniques, several theoretic foundations are provided as follows:

Definition 8 *An* **attribute descriptor** *is defined by a vector* $v = (w_1, w_2, \cdots, w_W)$*, where* w_j *denotes the term weight of term* j *and* W *denotes the number of all possible terms in our term dictionary.*

[1]Obviously, this assumption that all databases have the same set of attributes is impractical in reality. However, if there are heterogeneous attribute lists among different databases, this uniformed attribute list still can be produced by integrating those database attributes.

Apart from being used to represent a bibliographic record, attribute descriptors can also be used to capture the information about a cluster and a query. However, in the case of representing a bibliographic record, term frequencies are used as term weights (denoted by w_j's in the above definition).

Definition 9 *A* **bibliographic record** *is defined by a list of attribute descriptors,*

$$r = (vr_1, vr_2, \cdots, vr_l)$$

where each attribute descriptor vr_k represents the value for attribute A_k.

Definition 10 *A* **cluster** *consisting of a list of bibliographic records, $r_1, r_2, \cdots, r_{N_c}$, is defined by a binary tuple*

$$c = (N_c, D_c),$$

where D_c is a list of attribute descriptors

$$D_c = (vc_1, vc_2, \cdots, vc_l),$$

and each attribute descriptor vc_k in D_c captures the statistical information of the bibliographic records contained in cluster c with respect to attribute A_k, and

$$vc_k = vr_{1,k} + vr_{2,k} + \cdots + vr_{N_c,k}$$

where $vr_{j,k}$ denotes the kth attribute descriptor of r_j (the jth bibliographic records in the cluster).

In the above definition, D_c captures the representative content of all bibliographic records belonging to a cluster.

Definition 11 *The* **similarity between a bibliographic record $r(= (vr_1, \cdots, vr_l))$ and a cluster $c(= (N_c, (vc_1, \cdots, vc_l)))$**, *denoted by $SIM_{r,c}$, is defined as:*

$$SIM_{r,c} = \frac{1}{l} \sum_{k=1}^{l} SIM_{vr_k, vc_k} \tag{18}$$

where the SIM_{vr_k, vc_k} denotes the **similarity between two attribute descriptors**, *and is defined by:*

$$SIM_{vr_k, vc_k} = \frac{|vr_k \cdot vc_k|}{\sqrt{|vr_k|^2} \cdot \sqrt{|vc_k|^2}} \tag{19}$$

Intuitively, the similarity between a cluster and a bibliographic record is defined by averaging the similarities between the record and the cluster for all common attributes. Furthermore, the similarity between a cluster c and a bibliographic record r with respect to attribute A_k is defined as the cosine distance between attribute descriptor vr_k and vc_k.

Example 1:

Consider a database that contains two attributes A_1 = *title*, A_2 = *subject* and a term dictionary containing only three terms, i.e. *Information*, *Retrieval*, *Clustering* with term ids 1,2 and 3 respectively.

Let r_1 and r_2 be two bibliographic records shown below. They can be represented by $r_1 = ((1,0,1),(1,0,2))$[2] and $r_2 = ((1,1,0),(1,1,0))$, respectively.

record ids	*title*	*subject*
1	*information clustering*	*clustering, information clustering*
2	*information retrieval*	*information retrieval*

A cluster $c_1 = (10,((2,2,3),(2,1,5)))$ represents a cluster containing 10 bibliographic records. The two attribute descriptors $(2,2,3)$ and $(2,1,5)$ denote the weights of the three terms for the given two attributes(i.e. title and subject). Figure 13 shows the bibliographic records r_1, r_2 and cluster c_1 in a three-dimension space.

The similarity between the bibliographic record r_1 and cluster c_1 is:

$$\begin{aligned} SIM_{r_1,c_1} &= \frac{1}{2}\cdot\left(\frac{(1\times2+0\times2+1\times3)}{\sqrt{(1^2+0^2+1^2)\cdot(2^2+2^2+3^2)}}+\frac{(1\times2+0\times1+2\times5)}{\sqrt{(1^2+0^2+2^2)\cdot(2^2+1^2+5^2)}}\right) \\ &= 0.115 \end{aligned}$$

The similarity between the bibliographic record r_2 and cluster c_1 is:

$$\begin{aligned} SIM_{r_2,c_1} &= \frac{1}{2}\cdot\left(\frac{(1\times2+1\times2+0\times3)}{\sqrt{(1^2+1^2+0^2)\cdot(2^2+2^2+3^2)}}+\frac{(1\times2+1\times1+0\times5)}{\sqrt{(1^2+1^2+0^2)\cdot(2^2+1^2+5^2)}}\right) \\ &= 0.085 \end{aligned}$$

4.2.2 Proposed Database Clustering Techniques

Single Pass Clustering (SPC) and Rellocation Clustering (RC) are two straightforward clustering techniques used for text documents[FBY92]. To cluster bibliographic databases, the two techniques have been modified to cater for bibliographic records consisting of multiple text attributes. In addition, we have proposed a Constrained Clustering (CC) technique that generates for a bibliographic database a fixed number of clusters specified by the user. These three clustering techniques have been used with two different database ranking formulas given in Section 4.3.

[2]The two identical terms in the subject of the first bibliographic record are selected in the purpose to point out that attribute descriptors representing a bibliographic record could be non-binary vectors.

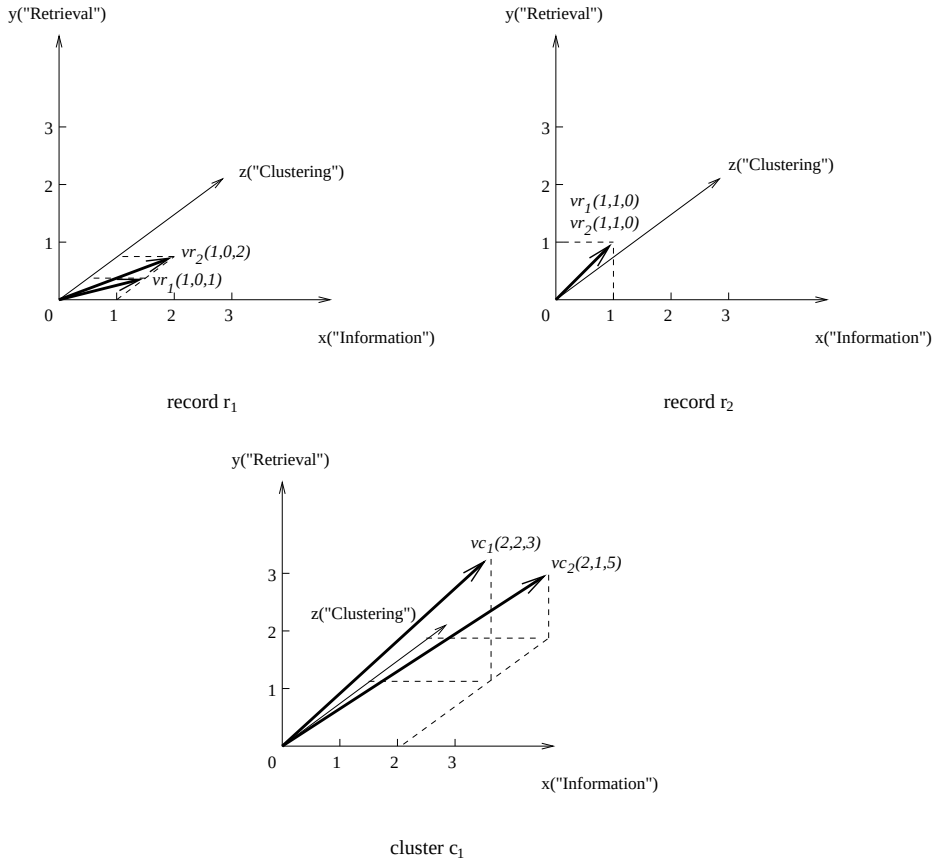


Figure 13: Representation of Records and Clusters

Single Pass Clustering Technique(SPC^C)

Single pass clustering technique is basically a greedy algorithm that always assigns a bibliographic record to the most similar cluster. Since each bibliographic record is read only once, SPC^C technique is efficient and easy to implement. Nevertheless, SPC^C technique requires a similarity threshold TH specified by the user. TH is used to determine if the similarity between a record and a cluster is large enough to assign the record to the cluster. When TH is small, each cluster can accommodate records that are less similar. Hence, a smaller number of clusters will be generated. The detailed clustering steps are given below and are depicted in Figure 14:

1. For each bibliographic record from the database, perform Steps (2) and (3).
2. Find the most similar cluster for the record among the existing clusters. The similarity measure between a bibliographic record and a cluster is given in Section 4.2.1.
3. (a) If no cluster has been created so far, or the similarity measures between the record and all existing clusters are smaller than the given threshold TH, a new cluster

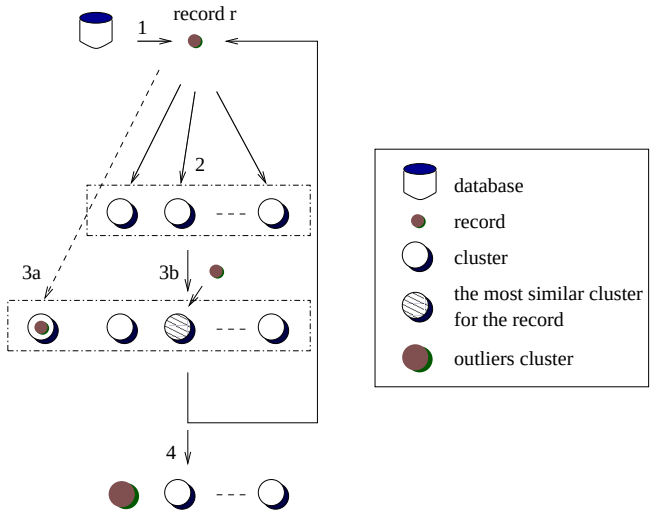


Figure 14: Single Pass Clustering

containing the record is created. (b) Otherwise, the record will be inserted into the cluster that is most similar.

4. All outlier clusters (clusters containing only 1 or 2 records) are combined into one at the end of the SPC^C clustering technique.

Although the single pass clustering technique has the advantage of simplicity, one might find that large clusters will often be generated by this clustering technique. Moreover, the performance of SPC^C technique depends on the order in which bibliographic records are processed.

Reallocation Clustering(RC^C)

Reallocation clustering[FBY92, GS98] operates by selecting an initial set of clusters followed by a series of iterations of re-assigning bibliographic records to the most similar clusters. Through the iterations, the cohesiveness among records in a cluster is improved. The following algorithm describes the steps required by the reallocation clustering technique for a bibliographic database. The algorithm is also illustrated in Figure 15.

1. Apply SPC^C to the database and use the clusters generated by SPC^C as the initial clusters.

2. For each bibliographic record from the database, perform Steps (3) and (4).

3. Find the most similar cluster for the record among the given clusters using the similarity measure given in Section 4.2.1.

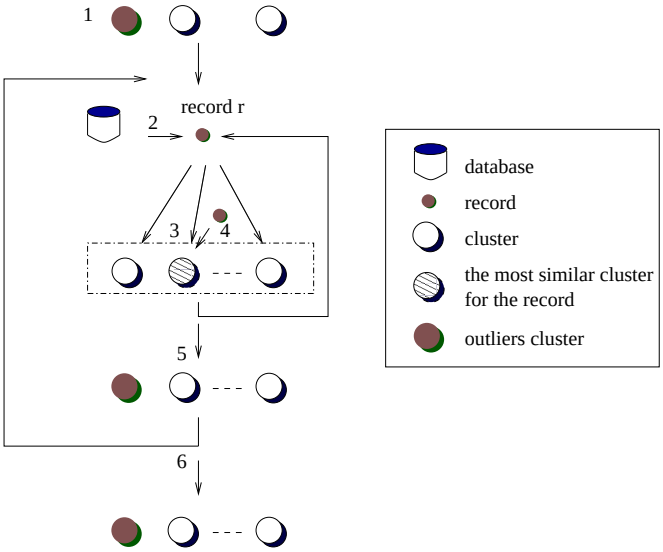


Figure 15: Reallocation Clustering

4. If the similarity measures between the record and all given clusters are smaller than the given threshold TH, the record will be inserted into the outliers' cluster. Otherwise, the record will be inserted into the cluster which is most similar.

5. After all records have been re-assigned, re-calculate the cluster vectors.

6. The resultant clusters of Step (5) are used as the input set of clusters for the next iteration of reallocation (i.e. Step (2) is performed again) until a specified number of iterations are completed.

In this technique, each record is assigned to a most similar cluster in every iteration. After a number of iterations, the set of clusters derived is expected to improve. However, it is difficult to decide how many iterations should be executed. For simplicity, we have chosen to apply 9 iterations in our experiments which will be described in Section 4.4. Like SPC^C, RC^C relies on a user specified threshold to control the number of clusters generated indirectly.

Constrained Clustering(CC^C)

In both SPC^C and RC^C, there is no control parameter that directly controls the storage requirement for the generated cluster information. The number of resultant clusters is controlled indirectly by the threshold TH. To overcome this shortcoming, we proposed the **Constrained Clustering** (CC^C) technique. CC^C is able to generate a fixed number β of clusters for each database where β is specified by the user. Like in the case of RC^C, CC^C requires an initial set of clusters to be first generated followed by iteratively improving the

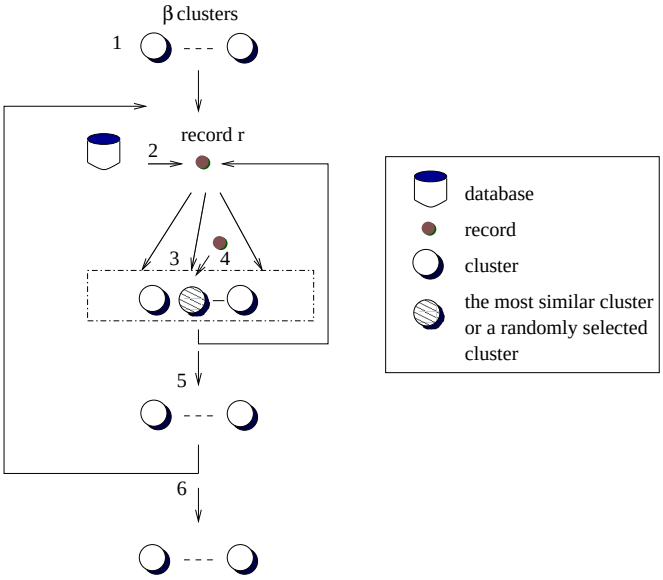


Figure 16: Constrained Clustering

similarity among records within the clusters. The algorithm, illustrated in Figure 16, is further described in detail as follows.

1. Use the first β largest clusters generated by SPC^C as the initial clusters.
2. For each bibliographic record from the database, perform Steps (3) and (4).
3. Find the most similar cluster for the record among the given clusters using the similarity measure given in Section 4.2.1.
4. If the similarity measures between the record and all given clusters are smaller than the given threshold TH, the record will be inserted into a randomly chosen cluster. Otherwise, the record will be inserted into the cluster which is most similar.
5. After all records have been processed, recalculate the cluster vectors.
6. The resultant clusters of Step (5) are used as the input set of clusters for the next iteration of reallocation (i.e. Step (2) is performed again) until a specified number of iterations are completed.

4.2.3 Discussions

Several relevant issues regarding to our proposed database clustering techniques for bibliographic databases are discussed as follows.

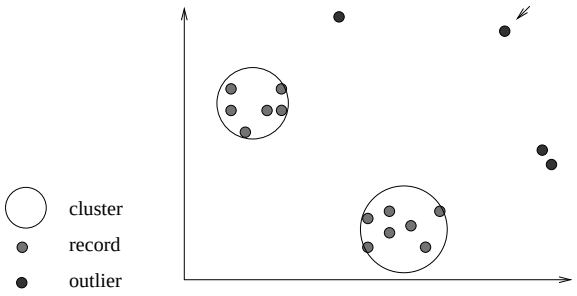


Figure 17: Outliers

Outliers

Outliers are records that are dissimilar to almost all other records as shown in Figure 17[GS98]. It is difficult to fit them into even the nearest cluster, i.e., the distance from the bibliographic record to its most similar cluster is much larger than the distance between any pair of records in that cluster. In this case, we have to decide whether outliers should be included into the most similar clusters (despite that they may not be similar enough) or to generate new clusters for them.

If we allow outliers to be included into individual clusters consisting of only one or two outlier records, large amount of storage resources will be required. On the other hand, if outliers are forced to be included into some clusters containing other records, the accuracy of clustering will be compromised. This becomes a trade-off between the clustering accuracy and the storage requirement of knowledge base. In SPC^C and RC^C clustering techniques for bibliographic database, all outliers are combined into a single cluster which is called outliers' cluster. In CC^C clustering technique, outliers' cluster is not included.

Less Significant Terms

Since the size of term dictionary is usually very large and the term frequency distribution is governed by the Zipf's Law, some clustering techniques [SYB97, SYB98] eliminate those *less significant terms(LST)* which have very small term frequencies. These less significant terms are eliminated on the basis that they have insufficient discriminatory power for objects to be clustered. We have also conducted experiments to evaluate the performance of clustering considering LST elimination(shown in Section 4.4).

4.3 Cluster-Based Database Ranking Formulas

In this section, two cluster-based database ranking formulas are given. They are defined based on the similarity between a given query and a database represented by a set of clusters.

Definition 12 *A* **query** *is defined to be a list of attribute descriptors,*

$$q = (vq_1, vq_2, \cdots, vq_l),$$

where each vector vq_k, which is an attribute descriptor for attribute A_k.

In each attribute descriptor, the weight for term j is 1 when term j is given in the query for the respective attribute. Otherwise, a term weight of 0 will be assigned.

Example 1(continued) : The query consisting of the following predicate: *subject = ("information" and "clustering")* can be represented by $q = (\vec{0}, (1, 0, 1))$.

Once a set of clusters have been generated for each database, we can apply the following two database ranking formulas to compute the rank of the database using the clusters information.

4.3.1 Cluster-based Database Ranking based on Estimated Result Sizes(ERS)

Given a query, this database ranking scheme computes the estimated query result size from a database by summing the estimated query result sizes returned by clusters belonging to the database. The ranking formula is formally defined below:

Definition 13 *The* **estimated result size(ERS)** *of a given query q from database db_i is defined as follows:*

$$G_{db_i,q} = \widehat{Size}_{(db_i,q)} = \sum_{n=1}^{|C|} \widehat{Size}_{(c_n,q)} \tag{20}$$

where $C = \{c_1, c_2, \cdots, c_{\beta_i}\}$ is a set of clusters generated for database db_i, β_i denotes the number of clusters generated for database db_i. The estimated result size of a given query q returned from a cluster c_n is defined as follows:

$$\widehat{Size}_{c_n,q} = N_{c_n} \cdot \prod_{\substack{k=1 \\ vq_k \neq \vec{0}}}^{|A|} \prod_{\substack{j=1 \\ w'_{j,k} \neq 0}}^{W} \frac{w_{j,k,n}}{N_{c_n}} \tag{21}$$

where $w'_{j,k}$ denotes the term weight of the jth dimension of the attribute descriptor vq_k for query q, $w_{j,k,n}$ denotes the term weight of the jth dimension of the attribute descriptor vq_k of cluster c_n, and N_{c_n} denotes the number of records that belong to the cluster c_n.

When the result size from a cluster for a given query is estimated, we assume that all attributes in the cluster are independently distributed and all terms in an attribute domain are also independently distributed. This assumption is important to the above formula that computes the estimated result size from a cluster. In the Formula 21, the predicates $vq_k \neq \vec{0}$ and $w'_{j,k} \neq 0$ indicate that only terms appearing in the query q and their corresponding terms appearing in cluster c_n will be considered in the computation.

4.3.2 Cluster-based Database Ranking based on Estimated Goodness Score(EGS)

Rather than estimating the query result size returned from each database, this ranking formula computes the goodness score of a database with respect to a given query.

Definition 14 *The* **estimated goodness score(EGS)** *of database db_i for a given query q is defined as follows:*

$$G_{db_i,q} = \sum_{n=1}^{|C|} G_{c_n,q} \tag{22}$$

where $C = \{c_1, c_2, \cdots, c_{\beta_i}\}$ is a set of clusters generated for database db_i.

The goodness score $G_{c_n,q}$, similar with that adopted by the DS technique in Section 3.2, of cluster c_n with respect to query q is defined by:

$$G_{c_n,q} = \prod_{\substack{k=1 \\ v q_k \neq \vec{0}}}^{|A|} \sum_{\substack{j=1 \\ w'_{j,k} \neq 0}}^{W} CVV_{j,k} \cdot w_{j,k,n} \tag{23}$$

where $CVV_{j,k}$ denotes the variance of $CV_{i,j,k}$'s, the Cue Validity *of term j, for attribute A_k across all databases, $w'_{j,k}$ denotes the weight of term j in attribute A_k for query q, $w_{j,k,n}$ denotes the term frequency of term j with respect to attribute A_k in cluster c_n.*

The formula to derive the CVV value of a term has been given in Section 3.2.

4.3.3 Discussions

Because the clusters of a database are derived based on all attributes of the database records, it's straightforward to compute the similarity between a cluster and a query that involves all attributes. Nevertheless, queries often do not involve all attributes. By applying the two database ranking formulas on one such query, we effectively compute the database ranks based on the cluster information in the dimensions where the attributes can be found in the query. This is analogous to using cluster information obtained by clustering the databases using the only attributes in the query.

4.4 Experiments

To evaluate the performance of database selection techniques that are built upon various combination of the three database clustering techniques and the two cluster-based database ranking formulas, a number of experiments have been conducted.

We conducted the experiments using the same experiment framework shown in Section 3.6. The difference is that we do not use training queries but use the summary knowledge about clusters. The experiments have been designed to answer a few questions about cluster-based database selection techniques:

- How do different database clustering techniques perform when the same database ranking formula is used?
- How do different database ranking formula perform when the same database clustering techniques is used?
- How do the cluster-based database selection techniques perform when generating different number of clusters?
- How do the cluster-based database selection techniques perform for bibliographic databases with different skewness in their content?
- How much storage requirement do our cluster-based database selection techniques need compared to non cluster-based database selection techniques?

4.4.1 Performance Evaluation

Since it is difficult or even impossible to find the ideal clusters for databases, we did not attempt to evaluate the performance of database clustering techniques. We only focus on the performance of our entire cluster-based database selection techniques.

The performance metric P defined in Section 3.6 is used to evaluate our cluster-based database selection techniques. Furthermore, we use the same set of 2000 test queries, which have been adopted in Section 3.6, to evaluate the performance of these techniques. Using this metric and these test queries, we can make comparison between the cluster-based database selection technique and the three previous techniques based on training queries.

4.4.2 Parameter Setting

The experiments are conducted by varying or fixing the following parameters which are used to perform the cluster-based database selection:

- M - the number of databases to be selected(M =1,2, ...,10)
- β - the number of clusters(only available for CC^C method, three values were selected $\beta = 20, 50, 100$)
- TH - the threshold of the similarity between a record and a cluster to decide whether to combine the record into the cluster. Five values were selected TH=0, 0.05, 0.1, 0.2, 0.4(available for SPC^C and RC^C methods, it only decides the initial clusters for CC^C method)
- Lp - the number of iterations is fixed to 9. (for RC^C and CC^C method)
- O - the minimum number of records in a normal cluster, O is fixed to $O = 3$ in this experiment which means that all clusters containing only two or one records will be considered as outlier clusters and be combined into one cluster
- L - the minimum result size for the test queries is fixed to 2. (see Section 3.6)
- Z_d - the database skew (Z_d = 0, 0.5, 1, 1.5, 2)

4.4.3 Experiments Finding

Figures 18 to 22 show the performance P against the number of databases to be selected(M) of our cluster-based database selection techniques. Six database selection techniques have been experimented by combining the 3 database clustering techniques (i.e., SPC^C, RC^C and CC^C) with the 2 database ranking formulas(i.e., ERS and EGS). For SPC^C and RC^C clustering techniques, the similarity threshold TH is a control parameter. By varying TH, different cluster-based database selection techniques perform differently. Note that, for CC^C clustering technique, the TH is only used in generating the initial set of clusters using SPC^C(e.g., $TH = 0.2$ and $\beta = 50$ for CC^C mean that CC^C clustering uses the first 50 largest clusters of SPC^C($TH = 0.2$) as the initial clusters).

The performance of database selection techniques using two ranking formulas based on SPC^C, (i.e., SPC^C-ERS and SPC^C-EGS)[3] with respect to four different similarity threshold TH are shown in Figure 18. Figure 19 shows the performance of RC^C-ERS and RC^C-EGS with respect to four TH values. Figures 20 to 22 show the performance of CC^C-ERS and CC^C-EGS using different initial clusters decided by TH and the number of clusters β fixed to 20, 50 and 100. In order to be compared with CC^C, the number of clusters generated for database db_0 by SPC^C and RC^C are shown in Figures 18 and 19[4].

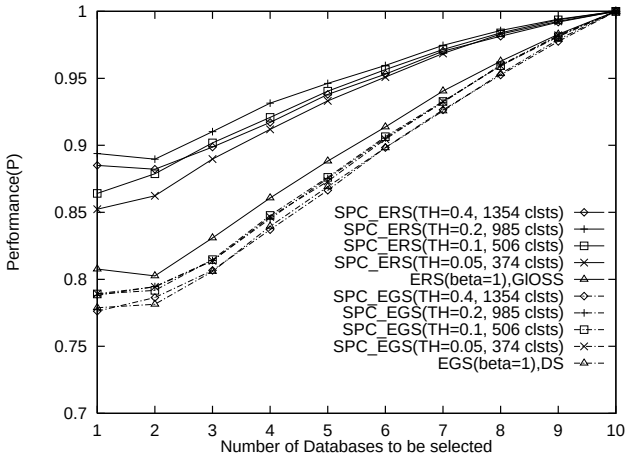


Figure 18: Performance of techniques using ERS and EGS based on $\mathrm{SPC}^C(Z_d{=}1)$

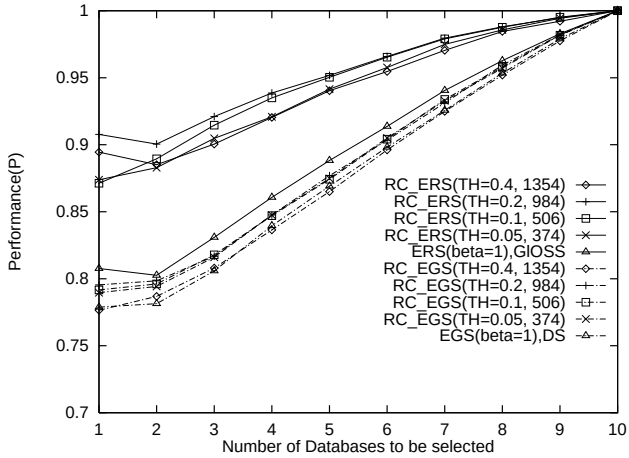


Figure 19: Performance of techniques using ERS and EGS based on $\mathrm{RC}^C(Z_d{=}1)$

In order to compare the performance of our proposed database selection techniques with that of others and our previous database selection techniques, the performance of GlOSS[GGMT94] and DS[XCLN98] are also shown in each of these figures as the baselines. Note that GlOSS database selection technique could be considered as an extreme case for our cluster-based database selection technique using only one cluster (i.e. $\beta = 1$) and ERS as the database ranking formula. On the other hand, DS technique could be considered as one extreme case of the cluster-based database selection technique using one cluster and EGS as the database ranking formula. Since all our proposed techniques outperform random database selection significantly, we do not show the performance of random database selection in these figures.

From the experiments conducted, we have several findings as described below:

- For SPC^C and RC^C, all database selection techniques with similarity threshold $TH = 0.2$ usually outperform the others.

  Explanation: When a large similarity threshold is chosen, the condition to combine records into clusters becomes stringent and the number of clusters increases. Moreover, the number of outliers will also increase. After combining the outliers into a outliers' cluster(see Section 4.2.2), a large number of records will be stored in the outliers' cluster. By having a large number of records stored in the outliers' cluster, the accuracy of clustering technique is reduced and it further worsens the performance our proposed database selection techniques. On the other hand, clustering techniques using small similarity threshold will generate a small number of clusters for each database. This might not reflect the exact distribution of database and will also compromise the database selection performance. In our experiments, it was shown that techniques using $TH = 0.2$ yield relatively good performance.

- Cluster-based database selection techniques using ERS significantly outperform those using EGS. The exact database clustering technique used does not even affect the performance of database selection techniques using EGS. This case occurs especially when a larger number of clusters were generated. When the number of clusters increases, database selection techniques using ERS outperforms those using EGS.

  Explanation: In [YL97], the GlOSS database selection technique is shown to perform better than the database selection technique using DS for a set of text documents. In our experiments, we notice that the same phenomenon also occurred in the case of databases containing multiple text attributes. The underlying reasons causing this are being investigated and will be reported in the final report.

[3] SPC^C-ERS denotes the database selection technique with ERS database ranking formula using SPC^C as the clustering method. This convention of naming database selection techniques will be used henceforth.

[4] As each database may have different number of clusters generated by SPC^C or RC^C, we only show the number of clusters for database db_0. Note that SPC^C and RC^C could generate slightly different numbers of clusters due to the possibility that some of the clusters may not be assigned any record during the reallocation phase in RC^C.

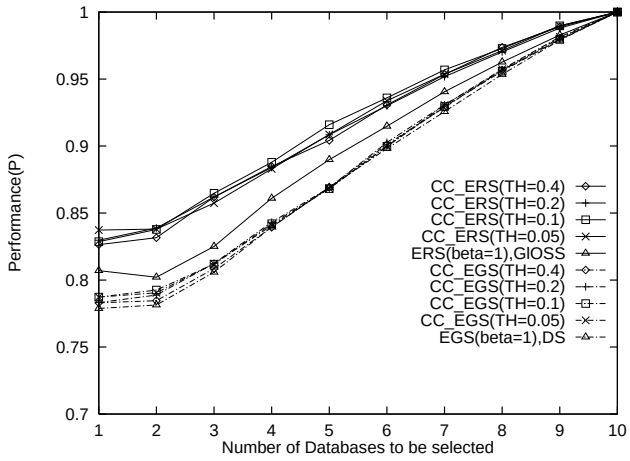


Figure 20: Performance of techniques using ERS and EGS based on CC^C with $\beta = 20(Z_d{=}1)$

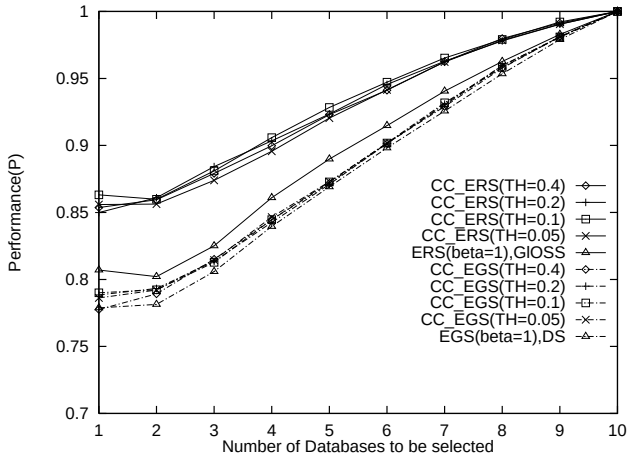


Figure 21: Performance of techniques using ERS and EGS based on CC^C with $\beta = 50(Z_d{=}1)$

- Database selection techniques using RC^C outperform those using SPC^C.

  Explanation: Reallocation method reassigns all records into clusters based on the actual distribution of the database after enough times of iterations. The clustering becomes more accurate after several times of iterations using RC^C than using SPC^C which only processes each record once.

- For CC^C technique, the choice of initial set of clusters is not important. For a given number of clusters, β, no matter what similarity threshold(TH) was chosen, the performance of our database selection techniques using CC^C is similar(see Figures 20 to 22).

In particular, the performance of database selection using CC^C clustering technique by changing β and fixing $TH = 0.2$ is depicted in Figure 23.

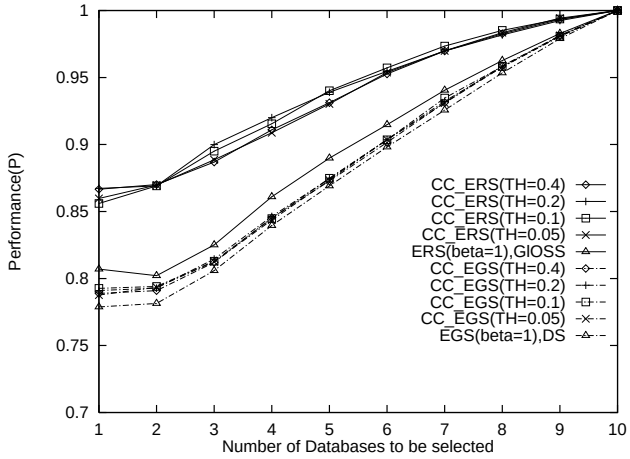


Figure 22: Performance of techniques using ERS and EGS based on CC^C with $\beta = 100(Z_d{=}1)$

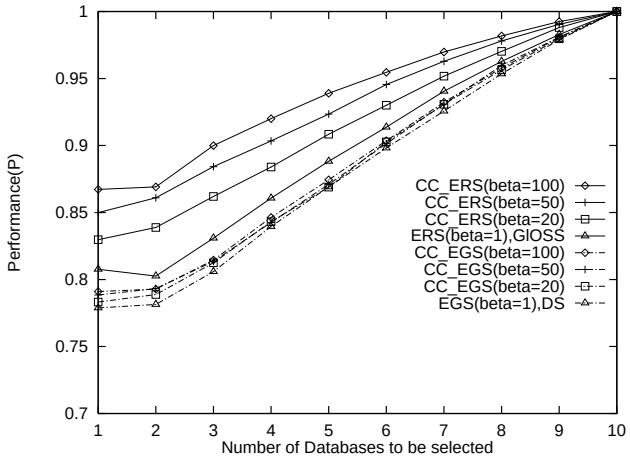


Figure 23: Performance of techniques using ERS and EGS based on CC^C as a function of the number of clusters, (TH=0.2, Z_d=1)

- As shown in Figure 23, for CC^C, the larger is the number of clusters, the more accurate is the performance of the database selection techniques.

To evaluate the performance of database selection of eliminating the less significant terms, we have conducted the experiment that applies the elimination of LST in the CC^C clustering technique(this procedure is denoted by ECC in the figure). The performance of this technique is shown in Figure 24. We found that:

- Less Significant Terms(LST) elimination is not useful for database selection.

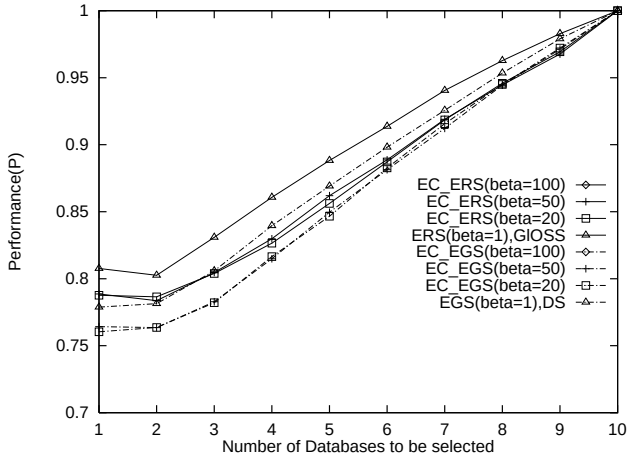


Figure 24: Performance of techniques using ERS and EGS based on CC^C considering LST (Less Significant Terms)(TH=0.2, Z_d=1)

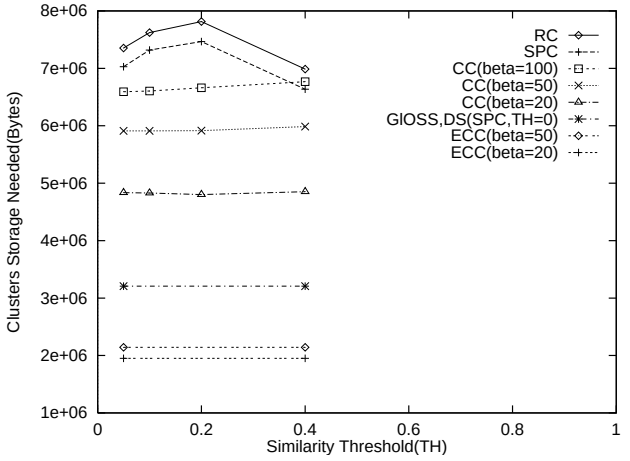


Figure 25: The storage space needed for database selection techniques(Z_d=1)

Furthermore, we investigate the storage requirement of database selection techniques using SPC^C, RC^C, CC^C(with β=20, 50,100), together with that of GlOSS and DS as the baselines[5]. Figure 25 shows the results.

- RC^C clustering technique has the largest storage requirement. The storage needed by CC^C can be adjusted by β and is highly lower than SPC^C and RC^C. We further find that CC^C(Constrained Clustering) needs relatively lower storage requirement and has acceptable performance.

The performance of our cluster-based techniques in different types of database skew values are shown in Figure 26 as well. The results are promising. Our proposed database

[5]The storage requirement of ECC are also shown as reference.

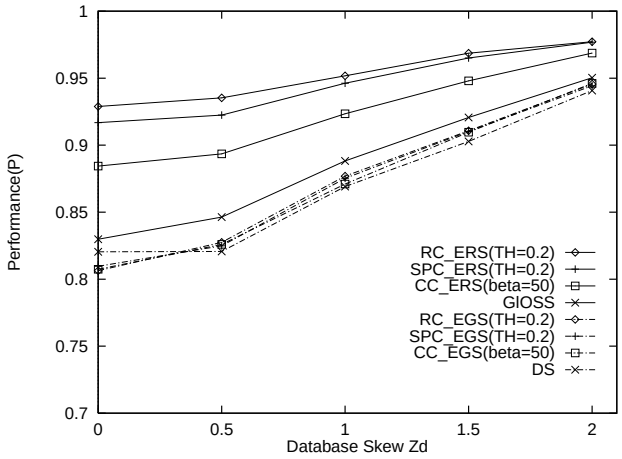


Figure 26: Performance of different cluster-based database selection techniques as a function of database skew value Z_d(M=5, TH=0.2 for SPC^C and RC^C, $\beta = 50$ for CC^C)

selection techniques RC^C-ERS, SPC^C-ERS and CC^C-ERS always outperform GlOSS and DS techniques in all database skew values. Even when the database skew is 0(i.e, the databases are randomly distributed), our proposed database selection techniques still have good performance. In particular, RC^C-ERS outperforms GlOSS by 12% when $Z_d = 0$ where it has about 5% increase when $Z_d = 2$. SPC^C-ERS and CC^C-ERS also have promising performance.

Chapter 5

CONCLUSIONS

Query routing is a common class of problems that involve selecting the appropriate information sources for a query to be evaluated, and merging the query results from the selected sources. In this report, two classes of database selection techniques have been proposed for routing queries to the relevant bibliographic databases on the Internet. We have proposed three database selection techniques based on training queries (TQS, TQRS and TQRG) and several cluster-based database selection techniques(SPC^C, RC^C and CC^C clustering, and ERS, EGS database ranking formulas). Unlike the previous database selection research that only focused on text document collections, our proposed techniques support databases and queries that involve multiple text attributes.

The TQS technique relies on a set of training queries and their actual result sizes to rank the databases relevant to a query. We also reexamined the GlOSS and DS techniques, two existing database selection techniques, which derive the ranks of databases from the tuple frequencies of terms in each database. By combining TQS and DS, the TQRS technique determines the database ranks from the tuple frequencies from the set of records sampled from each database using training queries. Similarly, the TQRG is derived from GlOSS and TQS. Through experiments, we have shown that the two new techniques, TQRG and TQRS, perform better than the TQS technique. Although the performance of GlOSS and DS techniques has been consistently good, they acquire complete tuple frequency statistics from all the bibliographic databases involved, Nevertheless, it is not much better than the TQRG and TQRS techniques which only require tuple frequency statistics from the training query results.

In the second class of database selection techniques, several cluster-based database selection techniques have been proposed. They are derived by combining three database clustering techniques with two database ranking formulas. Through experiments, we have shown that cluster-based database selection techniques outperform non cluster-based database selection techniques. However, clustering techniques need storage space more than their non

cluster-based counterparts. In cases where accuracy of database selection outweighs the storage overheads, cluster-based database selection techniques could be applied.

The other contribution of this project is that we have also proposed the use of Zipf-like distribution to generate database collections with controlled skew compositions of records from different categories. The Zipf-based approach[Gol84] to generate database collections with skew content can be applied to other database selection problems. Our experiments have shown our techniques yield better performance when the databases in the collection are highly skew in their content.

As part of our future work, we plan to pursue the following research directions:

- *Implementation:* We believe that these proposed database selection techniques can be applied to select bibliographic severs on the Internet. Hence, we plan to develop a query routing broker that incorporates the suitable database selection techniques for a distributed technical report collection. We will investigate the system issues involved in building such an intelligent broker and address them accordingly.

- *Database evolution:* Due to the time constraint, we have not investigated the database evolution issue. The three database selection techniques have to be extended to update their knowledge base as the databases evolve in their content. It is also important to keep the overhead of updating the knowledge bases low so that database selection can still be efficiently performed.

- *Experiments with other types of databases:* At present, our work has focused on bibliographic records which contain not many words in their attributes. As the proposed database selection techniques are generally applicable to any database with multiple text attributes, we plan to extend our experiments to other types of databases. In this case, it will be worthwhile to reexamine the performance when larger text have to be dealt with. Furthermore, the performance of our techniques with larger number of databases should also be investigated.

Appendix A

Symbol Table

Symbol	Description
D	a set of bibliographic databases
G	the set of databases to be selected in database selection
N	the number of member databases in D
M	the number of databases to be selected in database selection
db_i	database i in D
q	a query
s_i	the result size returned by db_i for query q
$KB(db_i)$	the knowledge base for database db_i
tq_j	the jth training query
$s_{i,j}$	the result size of tq_j returned by database db_i
p	the number of training queries
L_{tr}	the minimum result size that must be satisfied by training queries (L_{tr}=4 in our experiment)
$P(q)$	all selection predicates in a query q
$MP(q_1, q_2)$	the set of matching predicate pairs in query q_1 and q_2
$simp(p_1, p_2)$	the similarity measure of two matching predicates p_1 and p_2
$simq(q_1, q_2)$	the similarity measure between two queries q_1 and q_2
Q	a set of training queries
$\widehat{Size}_{(db_i,q)}$	the estimated result size of database db_i for a given query q
$G_{db_i,q}$	a goodness score to database db_i with respect to query q
$TF_{i,j,k}$	tuple frequency, the number of tuples(records) in database i containing term j in the attribute A_k
A	a set of text attribute
A_k	a attribute in A
$\|A_k\|$	the number of terms for attribute A_k
$CVV_{j,k}$	the variance of $CV_{i,j,k}$'s across all databases of term j for attribute A_k
N_i	the number of tuples in database db_i
$CV_{i,j,k}$	the *Cue Validity* of term j in database db_i for attribute A_k
$\overline{CV_{j,k}}$	the population mean of $CV_{i,j,k}$ over all databases for attribute A_k
$TF'_{i,j,k}$	the tuple frequency of term j for attribute A_k computed from the combined training query result for db_i
$CVV'_{j,k}$	the variance of $CV'_{i,j,k}$'s of term j for attribute A_k across all sample databases combined from the training query result
N'_i	the number of tuples in the combined training query result for db_i
V	a set of virtual categories
C_i	a set of records assigned to database db_i from a given category
Z_d	the database skew, the parameter used to apply a Zipf-like distribution
P	the performance measure
K	the number of test queries
B	the ideal database selection
P'	the mean-square rank error measure
$O_{i,j}$	the actual rank for database db_i for test query q_j
$R_{i,j}$	the rank of database db_i determined by our techniques for test query q_j
T	the number of training queries used to generate the knowledge base
L	the minimum result size for the test queries
TQS	database selection technique using Training Queries and their result Sizes
DS	database selection technique using Database Summary information
TQRS	database selection technique using Training Queries Result Summary information

Table 3: The Symbol Description Table for Database Selection Techniques Using Training Queries

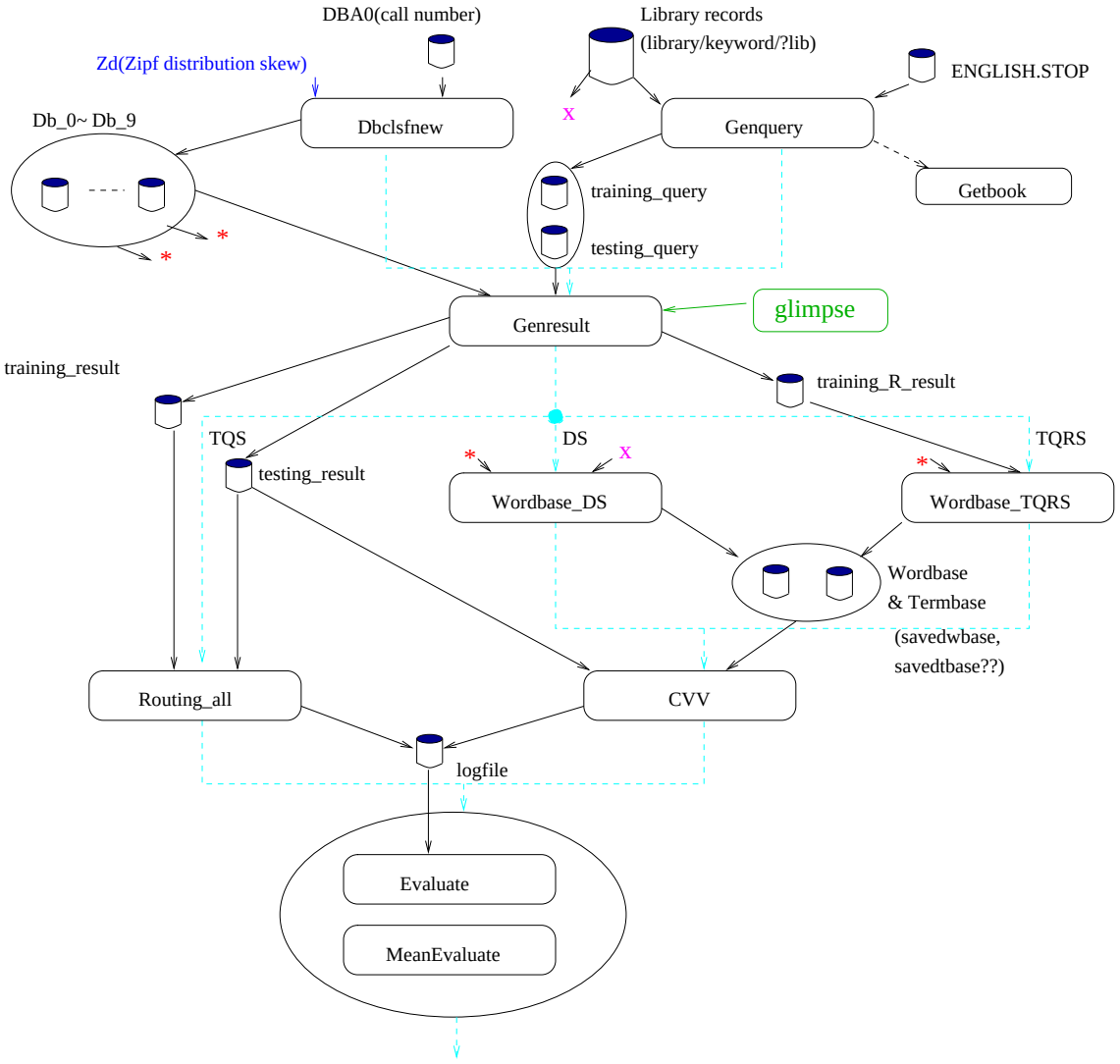
Symbol	Description
l	the number of attributes
W	the number of all possible terms in the text corpus representing the dimension of a text vector
v	a text vector, attribute descriptor
vr_k	the text vector with respect to attribute A_k for a record r
$vr_{j,k}$	the kth attribute descriptor for the jth record contained in a cluster
vc_k	the text vector with respect to attribute A_k for a cluster c
vq_k	the text vector with respect to attribute A_k for a query q
w_j	the term weight of the jth term
r	a record
c	a cluster
N_c	the number of tuples contained in cluster c
D_c	a list of text vectors contained in cluster c
$SIM_{r,c}$	the similarity between a record r and a cluster c
SIM_{vr_k,vc_k}	the similarity between two text vectors with respect to attribute A_k
$\vec{0}$	zero vector
c_n	the nth cluster
$\widehat{Size}_{c_n,q}$	the estimated result size from the cluster c_n for the give query q
$G_{c_n,q}$	the goodness score of cluster c_n with respect to query q
$w'_{j,k}$	the term weight of term j with respect to attribute A_k for query q
$w_{j,k,n}$	the term frequency of term j with respect to attribute A_k for cluster c_n
β	the number of clusters
β_i	the number of clusters generated for database db_i
TH	the threshold of the similarity between a record and a cluster
Lp	the number of iterations
LST	the maximum term frequency for less significant term
O	the minimum number of records in a normal cluster
SPC^C	single pass database clustering
RC^C	reallocation database clustering
CC^C	constrained database clustering
ERS	database ranking based on estimated result sizes
EGS	database ranking based on estimated goodness score

Table 4: The Symbol Description for Cluster-based Database Selection Techniques

Appendix B

Experiment Framework for Database Selection Techniques Based on Training Queries

Query Routing Experiment Framework



Program Description

Dbclsfnew: Generate databases given different Zd value
Genquery: Generate 8000 training queries & 2000 test queries
Genresult: Generate training query result and test query result
Routing_all: Routing phase for TQS technique
Wordbase_DS: Generate metadata info using all database records
Wordbase_TQRS: Generate metadata info using training query results
CVV: Routing phase for DS&TQRS techniques, with metadata info as the knowledge base to rank databases
Evaluate: Preformance measure P
MeanEvaluate: Mean-square rank error performance P'

Logical flow
Data flow(Input/Output)
Data file
Program

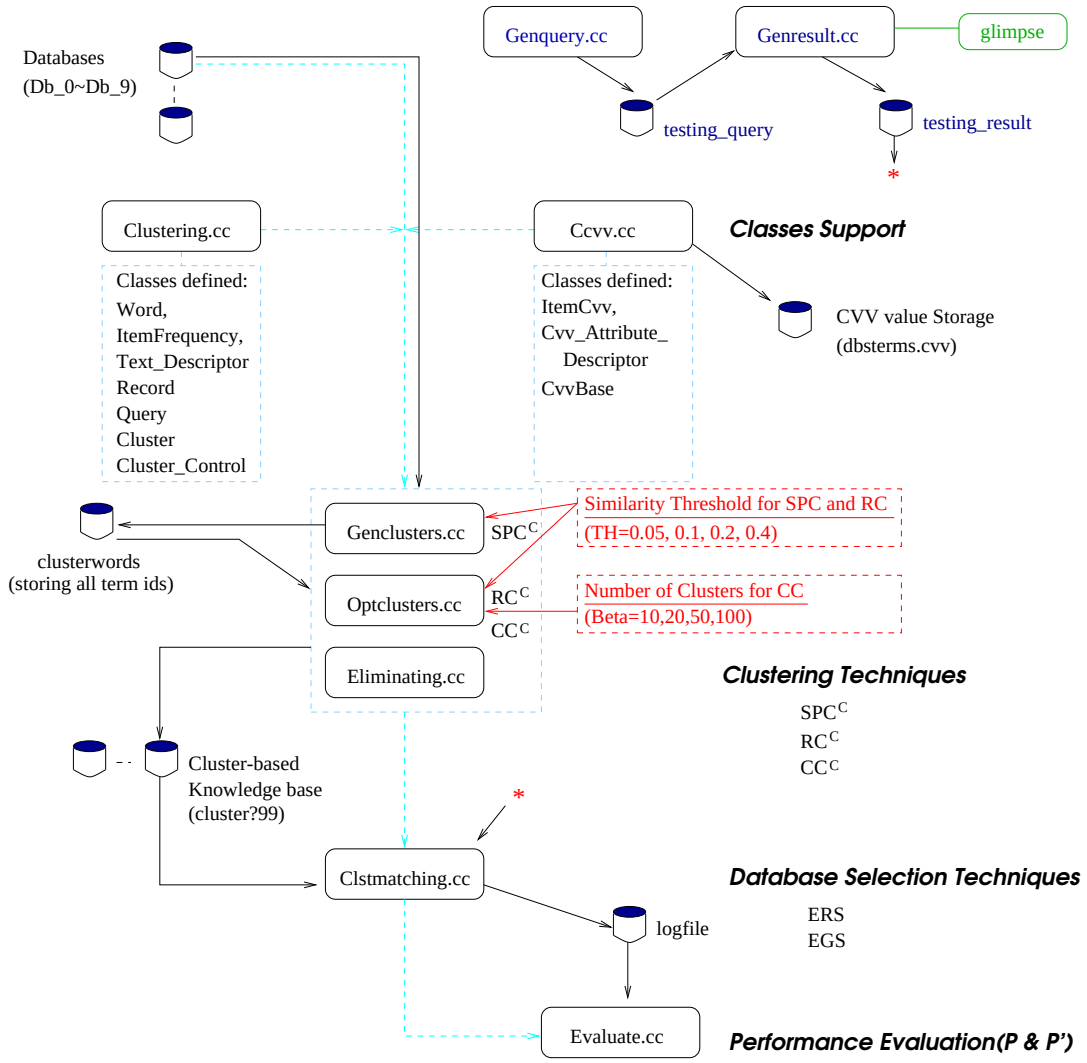
Filename: Framework.fig

Designer: Xu Jian Date: Jan 12, 1998

Appendix C

Experiment Framework for Cluster Based Database Selection Techniques

Cluster-Based Query Routing Experiment Framework



Program Description

Clustering.cc: Class implementation of database clustering techniques by defining several classes
Ccvv.cc: Cluster-based Cue Validity Variance related definition
Genclusters.cc: Building the initial clusters for each database
OptClusters.cc: Optimize the clustering using generic algorithm
Eliminating.cc: Eliminating less significant terms from clusters
Clstmatching.cc: Matching each test query with the clusters and estimating the result size from each database

Logical flow
Data flow(Input/Output)
Data file
Program

Filename: ClstFramework.fig

Designer: Xu Jian *Date: Aug 18, 1998*

Appendix D

Visualization of Term CVV Values and Term Frequencies

As a part of the result of our experiment, Figure 27 shows the CVV values of all terms across these ten databases with respect to *title* attribute calculated by DS technique. Figure 28 shows the term frequencies of all terms in database db_0 with respect to *title* attribute. The order of all term ids is decided by the order of the appearance of each term during processing the text attributes of every bibliographic record. Interestingly in Figure 27, there is a straight line in the middle of this figure which relates to several terms having cvv value 0.09. The reason is that there are a large number of terms only appearing in one of the databases once and the cvv value of them are identical.

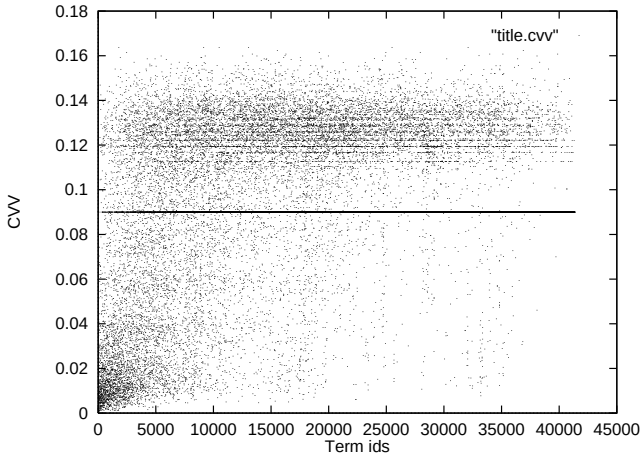


Figure 27: The term CVV values for attribute *title* calculated by DS technique

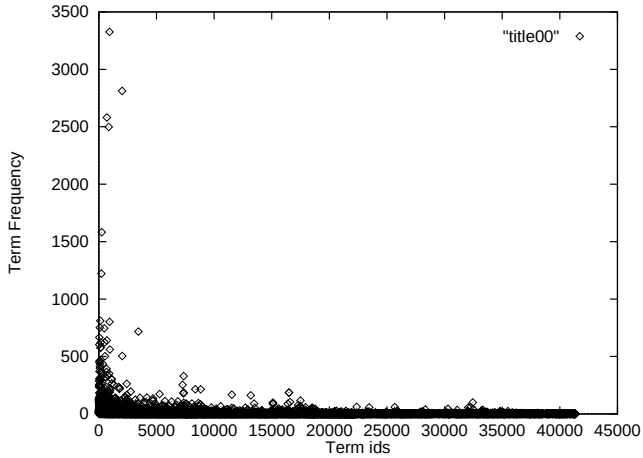


Figure 28: The term frequencies of terms in database db_0 with respect to attribute *title*

Bibliography

[Alt] Altavista. *http://www.altavista.digital.com.*

[BDF+97] D. Barbara, W. DuMouchel, C. Faloutsos, P.J. Haas, J.M. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng., V. Poosala, K.A. Ross, and K.C. Sevcik. The new jersey data reduction report. *Bulletin of the Technical Committee on Data Engineering*, 20(4), December 1997.

[CKPT92] D.R. Cutting, D.R. Karger, J.O. Pederson, and J.W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collection. In *Proceedings of ACM/SIGIR*, pages 318–329, 1992.

[CLC95] J.P. Callan, Z. Lu, and W.B. Croft. Searching Distributed Collections With Inference Networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, 1995.

[FBY92] W.B. Frakes and R. Baeza-Yates. *Information Retrieval : data structures & algorithms.* Englewood Cliffs, N.J., 1992.

[GCGM97] L. Gravano, C-C. K. Chang, and H. Garcia-Molina. STARTS: Stanford Proposal for Internet Meta-Searching. In *Proceedings of the ACM SIGMOD Conference*, pages 207–218, Tucson, Arizona, USA, May 1997.

[GGM95] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In *Proceedings of the 21st International Conference on Very Large Data Bases(VLDB'95)*, pages 78–89, Zurich, Switzerland, September 1995.

[GGMT94] L. Gravano, H. Garcia-Molina, and A. Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. In *Proceedings of the ACM SIGMOD*

International Conference on Management of Data, pages 126–137, Minneapolis, Minnesota, May 1994.

[Gol84] M.A. Golberg. *An Introduction to Probability Theory with Statistical Applications*. Plenum Press. New York and London, 1984.

[GP98] L. Gravano and Y. Papakonstantinou. Mediating and Metasearching on the Internet. *Bulletin of the Technical Committee on Data Engineering*, 21(2), June 1998.

[GS98] M. Goldszmidt and M. Sahami. A Probabilistic Approach to Full-Text Document Clustering. Technical Report ITAD-433-MS-98-044, SRI International, 1998.

[HLY93] K.A. Hua, Y-L. Lo, and H.C. Young. Considering Data Skew Factor in Multi-Way Join Query Optimization for Parallel Execution. *VLDB Journal*, 2(3):303–330, July 1993.

[KS97] D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 170–178, San Francisco, CA, 1997.

[LD97] S.H. Li and P.B. Danzig. Boolean Similarity Measures for Resource Discovery. *IEEE Transactions on Knowledge and Data Engineering*, 9(6), November/December 1997.

[Mea92] Charles T. Meadow. *Text Information Retrieval Systems*. San Diego : Academic Press, 1992.

[NCS] NCSTRL. *http://www.ncstrl.org*.

[Sal88] G. Salton. *Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Publishing Company, 1988.

[SHS96] M. Sahami, M. Hearst, and E. Saund. Applying the Multiple Cause Mixture Model to Text Categorization. In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, pages 435–443, San Francisco, CA, 1996.

[SYB97] M. Sahami, S. Yusufali, and M.Q.W. Baldonado. Real-time Full-text Clustering of Networked Documents. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, page 845, Menlo Park, CA, 1997.

[SYB98] M. Sahami, S. Yusufali, and M.Q.W. Baldonado. SONIA: A Service for Organizing Networked Information Autonomously. In *Proceedings of the 3rd ACM International Conference on Digital Libraries (DL'98)*, Pittsburgh, Pennsylvania, USA, June 1998.

[TGL+97] A. Tomasic, L. Gravano, C. Lue, P. Schwarz, and L. Haas. Data Structures for Efficient Broker Implementation. *ACM Transactions on Information Systems*, 15(3), July 1997.

[TVGJL95] G. Towell, E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird. Learning Collection Fusion Strategies for Information Retrieval. In *Proceedings of the 12th Annual Machine Learning Conference*, Lake Tahoe, July 1995.

[VF95] C.L. Viles and J.C. French. Dissemination of Collection Wide Information in a Distributed Information Retrieval System. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 12–20, 1995.

[XCLN98] J. Xu, Y.Y. Cao, E.P. Lim, and W.K. Ng. Database Selection Techniques for Routing Bibliographic Queries. In *Proceedings of the 3rd ACM International Conference on Digital Libraries (DL'98)*, Pittsburgh, Pennsylvania, USA, June 1998.

[Yah] Yahoo! *http://www.yahoo.com*.

[YL97] B. Yuwono and D.L. Lee. Server Ranking for Distributed Text Retrieval Systems on the Internet. In *Proceedings of the 5th International Conference on Database Systems for Advanced Applications (DASFAA '97)*, pages 41–49, Melbourne, Australia, April 1997.