$DPro$: A Probabilistic Approach for Hidden Web Database Selection Using Dynamic Probing

Victor Z. Liu, Richard C. Luo, Junghoo Cho, Wesley W. Chu

UCLA Computer Science Department
Los Angeles, CA 90095
{vicliu, lc, cho, wwc}@cs.ucla.edu

Abstract

An ever increasing amount of valuable information is stored in Web databases, "hidden" behind search interfaces. To save the user's effort in manually exploring each database, *metasearchers* automatically select the most relevant databases to a user's query [2, 5, 16, 21, 26]. Existing methods use a pre-collected summary of each database to estimate its "relevancy" to the query, and return the databases with the highest estimation. While this is a great starting point, the existing methods suffer from two drawbacks. First, because the estimation can be inaccurate, the returned databases are often wrong. Second, the system does not try to improve the "quality" of its answer by contacting some databases on-the-fly (to collect more information about the databases and select databases more accruately), even if the user is willing to wait for some time to obtain a better answer. In this paper, we introduce the notion of *dynamic probing* and study its effectiveness under a probabilistic framework: Under our framework, a user can specify how "correct" the selected databases should be, and our system automatically contacts a few databases to satisfy the user-specified correctness. Our experiments on 20 real hidden Web databases indicate that our approach significantly improves the correctness of the returned databases at a cost of a small number of database probing.

1 Introduction

An ever increasing number of information on the Web is available through search interfaces. This information is often called the *Hidden Web* or *Deep Web* [1] because traditional search engines cannot index them using existing technologies [10, 23]. Since the majority of Web users rely on traditional search engines to discover and access information on the Web, the Hidden Web is practically inaccessible to most users and "hidden" from them. Even if users are aware of a certain part of the Hidden Web, they need to go through the painful process of issuing queries to all potentially relevant *Hidden Web databases*[1] and investigating the results manually. On the other hand, the information in the Hidden Web is estimated to be significantly larger and of higher quality than the "Surface Web" indexed by search engines [1].

keyword	document frequency in db_1	document frequency in db_2
cancer	10,000	5,000
kidney	7,000	10,000
breast	2,000	3,500
liver	200	4,000

Figure 1: (Keyword, document frequency) table. Document frequency of a keyword in db is the number of documents in db that use the keyword

In order to assist users accessing the information in the Hidden Web, recent efforts have focused on building a *metasearcher* or a *mediator* that automatically selects the most relevant databases to a user's query [2, 5, 14, 15, 16, 18, 21, 24, 25, 26]. In this framework, the metasearcher maintains a summary or statistics on each database, and consults the summary to *estimate* the relevancy of each database to a query. For example, Gravano et al. [14, 16] maintain (keyword, document frequency) pairs to estimate the databases with the most number of matching documents. We illustrate the basic idea of the existing approaches using the following example.

Example 1 A metasearcher mediates two Hidden-Web databases, db_1 and db_2. Given a user's query q, the goal of the metasearcher is to return the database with the most number of matching documents. The metasearcher maintains the (keyword, document frequency) table shown in Figure 1.[2] For example, the first row shows that 10,000 documents in db_1 contain the word "cancer" while 5,000 documents in db_2 contain "cancer." We assume that each of db_1 and db_2 has a total of 20,000 documents.

Given a user query "breast cancer," the metasearcher may select the database with more matching documents in the following way: From the summary we know that $\frac{2,000}{20,000}$ fraction of the documents in db_1 contain the word "breast" and $\frac{10,000}{20,000}$ of them contain the word "cancer." Then, assuming that the words "breast" and "cancer" are independently distributed, db_1 will have $20,000 \cdot \frac{2,000}{20,000} \cdot \frac{10,000}{20,000} = 1,000$ documents with both words "breast" and "cancer." Similarly, db_2 will have $20,000 \cdot \frac{3,500}{20,000} \cdot \frac{5,000}{20,000} = 875$ matching documents. Based on this estimation, the metasearcher returns db_1 to the user. □

[1] We call a collection of documents accessible through a Web search interface as a Hidden-Web database. PubMed (`http://www.ncbi.nlm.nih.gov/entrez/query.fcgi`) is one example.

[2] References [18] explain in detail how we may construct this table from hidden Web databases.

In this paper, we improve upon this existing framework by introducing the concept of *probabilistic correctness* and *dynamic probing* for Hidden Web database selection. One of the main weaknesses of the existing method is that the selected databases are often not the most relevant to the user's query, because the relevancy of a database is *estimated* based on a pre-collected summary. For instance, in the above example, the word "breast" and "cancer" may not be independently distributed, and db_2 may actually contain more matching documents than db_1. Given a wrong answer, the user ends up wasting a significant amount of time on the irrelevant databases. Recent study shows that investigating irrelevant Web pages is a major cause of user's wasted time on the Web [3].

One way of addressing this weakness is to issue the user's query q to all the databases that the metasearcher mediates, and select the best ones based on the actual result returned by each database. For instance, the metasearcher may issue the query "breast cancer" to both db_1 and db_2 in the above example, obtain the number of matching documents reported by them and select the one with more matches. While this approach can improve the "correctness" of database selection, its huge network and time overhead makes it impractical when metasearchers mediate a large number of Hidden-Web databases (often thousands of them [1]).

In this paper, we develop a probabilistic approach to use *dynamic probing* (issuing the user query to the databases on the fly) in a systematic way, so that the correctness of database selection is significantly improved while the metasearcher contacts the minimum number of databases. In our approach, the user can specify the desired correctness of database selection (e.g., "more than 9 out of the 10 selected databases should be the actual top 10 databases"), and the metasearcher decides how many and which databases to contact based on the user's specification. Informally, we may consider the user-specified correctness as a "knob:" When the user does not care about the answer's correctness, our approach becomes identical to the existing ones (no dynamic probing). As the user desires higher correctness, our approach will contact more databases. Our experimental results reveal that dynamic probing often returns the best databases with a small number of probing.

Dynamic probing of Web databases introduces many interesting challenges. For example, how can we guarantee a certain level of correctness? How can we maximize the correctness with the minimal number of dynamic probing? Which databases should we probe? This paper studies these problems using a probabilistic approach.

Our solution is based on the following observations: Although the actual relevancy of a Web database may deviate from an initial inaccurate estimation, the way it deviates follows a probabilistic distribution that can be observed. Such a distribution usually centers around the estimated relevancy value. If we roughly know this actual relevancy distribution for each database, then we can "guess" how likely we have selected the actual top-k databases using these distributions. Furthermore, by probing a few databases, we can obtain their *actual* relevancy values and we can select top-k databases with higher confidence. Our task of dynamic probing thus becomes using the minimum number of probing to accomplish the user-specified correctness level. We will formalize these notions, e.g. the probabilistic distribution and the correctness of an answer, in Section 2. Overall, we believe our paper makes the following contributions:

1. **A probabilistic model for relevancy estimation:** With the probabilistic model, we can quantify the correctness of database selection. (Section 2)
2. **Using dynamic probing to increase the correctness of database selection:** We keep on probing till the certainty exceeds a user-specified level. (Section 3)
3. **Probing strategies:** Our optimal strategy uses the minimum number of database probing to reach the required level of certainty. (Section 3.1) We also present a greedy strategy that can identify top-k databases at reasonable computational complexity. (Section 3.2)
4. **Experimental validation:** We validate our algorithms using real Hidden Web databases, under various experimental settings. (Section 5) The results reveal that dynamic probing significantly improves the correctness of database selection with a reasonably small number of probing. For example, with a single probing, we can improve the correctness of an answer by 70% in certain cases.

2 A Probabilistic Approach for Dynamic Probing

To select the most relevant databases for a query and make our selection as correct as possible, we need to fully understand the "relevancy" of a database to a query, and the "correctness" of a set of selected databases. In this section, we first define the relevance metric of a database. We then introduce the notion of expected correctness for a top-k answer set. Finally, we explain the cost model for dynamic probing.

2.1 Database relevancy and probing

Relevancy of a database Intuitively, we consider a database relevant to a query if the database contains enough documents pertinent to the query topic. The following are two possible definitions that reflect this notion of relevancy.

- *Document-frequency-based* A database is considered the most relevant if it contains the highest number of matching documents [14, 16]. This number of matching documents is referred to as the *document frequency* of the query in the database.
- *Document-similarity-based* A database is considered the most relevant if it contains the most similar document(s) to the query [15, 21, 25]. Query-document similarity is often computed using the standard *Cosine* function [22].

Relevancy estimation A metasearcher has to estimate the approximate relevancy of a database to a query using a pre-collected summary. Note that this estimate may or may not be the same as the actual relevancy of the database. We refer to the estimated relevancy of a database db to a query q as $\tilde{r}(db, q)$.

To make our later discussion concrete, we now briefly illustrate how we may estimate the relavancy of a database under the document-frequency-based metric [14]. Note, however, that our framework is independent of the particular relevancy metric and the relevancy estimator used by a metasearcher. Our approach can be used for any relevancy metric and estimator combination.

In [14, 16], Gravano et al. compute $\tilde{r}(db, q)$ by assuming that the query terms $q = \{t_1, ..., t_m\}$ are independently distributed in db. Using their independence estimator, $\tilde{r}(db, q)$ can be computed as follows:

$$\tilde{r}(db, q) = |db| \cdot \prod_{t_i \in q} \frac{r(db, t_i)}{|db|} \quad (1)$$

where $|db|$ is the size of db and $r(db, t_i)$ is the number of documents in db that use t_i. Note that Eq.(1) assumes that $r(db, t_i)$ is available to the metasearcher for every term t_i and every database db. In practice, however, a hidden web database seldom exports such an exhaustive content summary to the metasearcher. Reference [18] proposes an approximation method to guess the $r(db, t_i)$ values for all query terms.

Database probing We define *probing* a database as the operation of issuing a particular query to the database and gathering the necessary information to evaluate its *exact* relevancy to the query. Depending on the relevancy metric, we need to collect different information during probing. For example, under the document-frequency-based metric, we need to collect the number of matching documents from the probed database, while under the document-similarity-based metric, we need to collect the similarity value of the most similar document(s) in the probed database.

For most existing Hidden Web databases, we note that it is possible to get their exact relevancy through simple operations. For instance, many Hidden Web databases report the number of matching documents in their answer page to a query, so we can easily compute their exact document-frequency-based relevancy. Also, under the document-similarity-based metric, we may download the first document that a Hidden Web database returns, and then analyze its content to compute its cosine similarity. In the remainder of this paper, we refer to the exact relevancy of a database db to a query q as $r(db, q)$. Thus, after probing db, its estimated relevancy $\tilde{r}(db, q)$ becomes $r(db, q)$.

2.2 Correctness metric for the top-k databases

Our goal is to find the k databases that are most relevant to a query. We represent this set of correct top-k answers as DB^{topk}. We refer to the set of k databases selected by a particular selection algorithm as DB^k. We may define the correctness of DB^k compared to DB^{topk} in one of the following ways.

- *Absolute correctness*: We consider DB^k is "correct" only when it contains all DB^{topk}.

  Definition 1 The *absolute correctness* of DB^k compared to DB^{topk} is

  $$Cor_a(DB^k) = \begin{cases} 1 & \text{if } DB^k = DB^{topk} \\ 0 & \text{otherwise} \end{cases} \quad \square$$

- *Partial correctness*: We give "partial credit" to DB^k if it contains some of DB^{topk}.

  Definition 2 The *partial correctness* of DB^k compared to DB^{topk} is

  $$Cor_p(DB^k) = \frac{|DB^k \cap DB^{topk}|}{k} \quad \square$$

  In this definition, the correctness value of a top-5 answer set is 0.4 if it contains 2 of the actual top 5 databases.

We study both of these metrics in this paper. For reader's convenience, we summarize the notation that we have introduced in Figure 2. Some of the symbols will be discussed later.

Symbol	Meaning
DB	$\{db_1, ..., db_n\}$, the total set of databases
q	The user's query
k	The number of top databases asked by the user
$r(db, q)$	The actual relevancy of db for q
$\tilde{r}(db, q)$	The estimated relevancy of db for q
DB^k	A set of k databases selected by a particular algorithm, $DB^k \subseteq DB$
DB^{topk}	The set of correct top-k databases
$Cor_a(DB^k)$	Absolute correctness metric for DB^k
$Cor_p(DB^k)$	Partial correctness metric for DB^k
DB^P	The set of databases that have already been probed
DB^U	The set of databases that have not been probed, i.e. $DB - DB^P$
PRD	Probabilistic Relevancy Distribution
$P(r(db, q) \leq \alpha \mid \tilde{r}(db, q) = \beta)$	The probability of $r(db, q)$ being lower than α, given the relevancy estimation $\tilde{r}(db, q) = \beta$. This probability is given by the PRD. α and β are specifi c relevancy values
$E[Cor(DB^k)]$	The expected correctness of DB^k, here Cor can be Cor_a or Cor_p
t	The user-specifi ed threshold of the answer's expected correctness
c	The cost of probing a database
$\mathsf{ECost}(DB^U)$	The expected probing cost on the set of unprobed databases, DB^U
$err(r, \tilde{r})$	The error function computing the difference between $r(db, q)$ and $\tilde{r}(db, q)$

Figure 2: Notation used throughout the paper

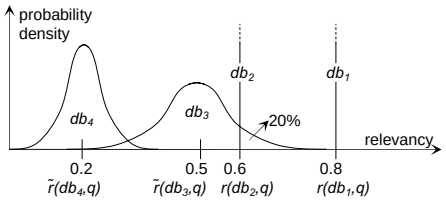
2.3 Probabilistic relevancy distribution and expected correctness

While we may estimate the relevancy of a database db to a query q, $\tilde{r}(db, q)$, using existing relevancy estimators, we do not know the exact $r(db, q)$ value until we actually probe db. Therefore, we may model $r(db, q)$ to follow a probabilistic distribution that (hopefully) centers around the $\tilde{r}(db, q)$ value. We refer to this distribution as a *Probabilistic Relevancy Distribution*, or *PRD*. In Figure 3(a), we show example PRDs for four databases, $db_1, \ldots, db_4$. The horizontal axis in the figure represents the actual relevancy value of a database and the vertical axis shows the probability density that the actual relevancy is at the given value. For instance, for db_3, the estimated relevancy is 0.5, and the actual relevancy lies between 0.2 and 0.75. (We explain the impulses for db_1 and db_2 shortly.) Formally, a PRD tells us the probability that $r(db, q)$ is lower than a certain value α given the relevancy estimate $\tilde{r}(db, q)$ equals to β: $P(r(db, q) \leq \alpha \mid \tilde{r}(db, q) = \beta)$. In Section 4 we explain how we can obtain a PRD by issuing a small number of sample queries to a database. For now we assume that the metasearcher knows the PRD of every database.

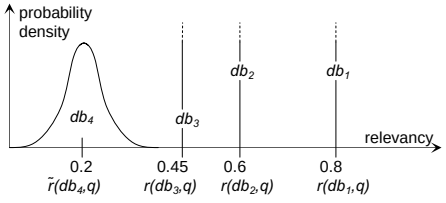
Note that after probing db, the $r(db, q)$ value is known. Thus the PRD for $r(db, q)$ changes from a broad distribution to an impulse function. For example, in Figure 3(a), we assume db_1 and db_2 have already been probed, so their PRDs have become impulses at their correct relevancy values, 0.8 and 0.6, respectively. In the middle of a dynamic-probing process, therefore, we have impulse PRDs for the probed databases, and regular PRDs for the rest.

We now illustrate how we can use the PRDs to estimate the probability that a top-k answer DB^k is correct.

Example 2 We assume the situation shown in Figure 3(a): db_1 and db_2 have already been probed and their relevancy values



(a) after probing db_1 and db_2



(b) after probing db_1, db_2 and db_3

Figure 3: The Probabilistic Relevancy Distribution of different databases at various stages of probing

are 0.8 and 0.6, respectively. We do not know the exact relevancy values for db_3 and db_4, but the PRD of db_3 indicates that $r(db_3, q) \geq r(db_2, q) = 0.6$ with 20% probability. We use the absolute correctness metric of an answer set.

Now suppose the user wants the metasearcher to return the top-2 databases. In this scenario, if we return $\{db_1, db_2\}$, our answer is correct ($Cor_a(\{db_1, db_2\}) = 1$) with 80% probability, because $r(db_3, q) < r(db_2, q)$ with 80% probability (in which case $\{db_1, db_2\}$ are the actual top-2 databases).[3] With the remaining 20% probability, $r(db_3, q)$ may be larger than $r(db_2, q)$, so our answer $\{db_1, db_2\}$ is wrong ($Cor_a(\{db_1, db_2\}) = 0$) with 20% probability. Therefore, the *expected correctness* of the answer $\{db_1, db_2\}$, is $1 \cdot 0.8 + 0 \cdot 0.2 = 0.8$. □

The expected correctness in Example 2 can be better understood on a statistical basis. For example, if the user issues 1,000 queries to a metasearcher, and the metasearcher return DB^k such that its expected correctness is greater than 0.8 for every query, then the user gets correct answers for at least 800 queries.

We now illustrate how a user may use the expected correctness to specify the "quality" of the answer and how the metasearcher can use dynamic probing to meet the user's specification.

Example 3 Still consider the situation in Figure 3(a). After probing db_1 and db_2, the metasearcher knows that the expected correctness of $\{db_1, db_2\}$ is 0.8. If the user only requires 0.7 expected correctness, the metasearcher can stop probing and return $\{db_1, db_2\}$. If the user's threshold is 0.9, the metasearcher has to probe more databases. Suppose the metasearcher picks db_3 for probing. The resulting PRDs are shown in Figure 3(b). Now the metasearcher knows that db_3 and db_4 are definitely smaller than db_2, and $\{db_1, db_2\}$ must be the correct answer. Therefore the expected correctness of $\{db_1, db_2\}$ is 1 (which exceeds the user's threshold, 0.9). As a result, the metasearcher can stop probing and return $\{db_1, db_2\}$. □

The above example shows that we can consider the expected correctness as the "knob" that the user can turn so as to control the result "quality." Given the user's expected correctness specification, the metasearcher keeps on probing databases till it finds a DB^k that exceeds the user-specified threshold.

To help our discussion, we refer to the set of databases that have been probed during this process as DB^P and the set of unprobed databases as DB^U. Note that the returned databases DB^k may or may not be the same as the probed databases DB^P. In particular, DB^k may contain a database db that may have not been probed ($db \notin DB^P$). As long as the metasearcher is confident that $r(db, q)$ is higher than those of others, it is safe to return db as part of DB^k.

From the example, it is clear that we should be able to compute the expected correctness for DB^k given PRDs of the databases. We use the notation $E[Cor_a(DB^k)]$ and $E[Cor_p(DB^k)]$ to refer to the expected correctness of DB^k under the absolute and partial correctness metric, respectively. When we do not care about a particular correctness metric, we use the notation $E[Cor(DB^k)]$.

According to our Cor_a and Cor_p definitions, the expected correctness can be computed as:

$$\begin{aligned} &E[Cor_a(DB^k)] \\ &= 1 \cdot P(DB^k = DB^{topk}) + 0 \cdot P(DB^k \neq DB^{topk}) \\ &= P(|DB^k \cap DB^{topk}| = k) \end{aligned} \tag{2}$$

$$\begin{aligned} &E[Cor_p(DB^k)] \\ &= \sum_{1 \leq i \leq k} \frac{i}{k} \cdot P(|DB^k \cap DB^{topk}| = i) \end{aligned} \tag{3}$$

The following theorems tell us how to compute the expected absolute correctness, $E[Cor_a(DB^k)]$, and the expected partial correctness, $E[Cor_p(DB^k)]$, using the PRD of each database. We label the databases in DB^k as db_1, db_2, ..., db_k, and label the databases in $DB - DB^k$ as db_{k+1}, ..., db_n. Let $f_j(x_j)$ be the Probability Density Function derived from db_j's PRD ($1 \leq j \leq n$), and x_j be one possible value of $r(db_j, q)$.

Theorem 1 *Assuming that all databases operate independently,*

$$E[Cor_a(DB^k)] = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \left(\prod_{db \in DB - DB^k} P(r(db, q) < \min(x_1, ..., x_k)) \right) \times \left(\prod_{1 \leq j \leq k} f_j(x_j) \right) dx_1 ... dx_k$$

where $\min(x_1, ..., x_k)$ *is the minimum relevancy value among all the* $db_j \in DB^k$. □

Proof See Appendix ■

[3] Note that $r(db_4, q)$ is always smaller than $r(db_1, q)$ and $r(db_2, q)$.

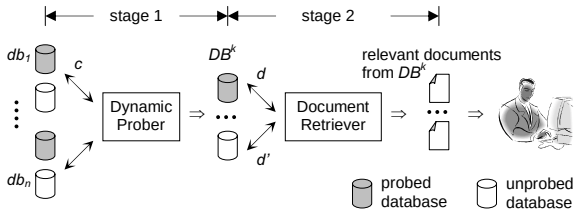


Figure 4: Two stage cost models

Theorem 2 *Assuming that all databases operate independently,*

$$E[Cor_p(DB^k)] =$$

$$\sum_{1\le i\le k} \frac{i}{k} \cdot \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \left(\sum_{\substack{DB^{k-i} \subseteq \\ DB - DB^k}} \prod_{db \in DB^{k-i}} P(r(db,q) > i\_highest(x_1,...,x_k)) \cdot \prod_{\substack{db' \in (DB - \\ DB^k - DB^{k-i})}} P(r(db',q) < i\_highest(x_1,...,x_k)) \right) \times \left(\prod_{1\le j\le k} f_j(x_j) \right) dx_1 ... dx_k$$

where $i\_highest(x_1,...,x_k)$ is a function that computes the i_{th} highest relevancy value among all the $db_j \in DB^k$. □

Proof See Appendix ■

2.4 Two-stage cost models

When a user interacts with a metasearcher, his eventual goal is to retrieve a set of relevant documents. Therefore, the overall metasearching process can be separated into two stages as shown in Figure 4. In the first stage, the *dynamic prober* finds an answer set DB^k by probing a few databases. In the second stage, the *document retriever* contacts each selected database, retrieves the relevant documents and returns them to the user. In measuring the cost of our metasearching framework, we may use one of the following metrics:

- *Probing cost model:* We only consider the cost for the probing stage, ignoring the cost for the document-retrieval stage. We assume that the probing cost of a single database is c and is identical for every database. (It is straightforward to extend our model to the case where the probing cost for each database is different.) Since the dynamic prober does $|DB^P|$ number of probing in the first stage, $|DB^P| \cdot c$ is the cost under this model.
- *Probing-and-Retrieval (PR) cost model:* We also consider the cost for the document retrieval stage. The cost for the retrieval stage may depend on whether a selected database was probed or not in the first stage. For example, suppose the dynamic prober returns $\{db_1, db_2\}$ as the top-2 databases after probing db_1 (but not db_2). If the dynamic prober has retrieved the top ranking documents of db_1 during its probing of db_1,[4] then the matasearcher may have "cached" the retrieved documents, so that it will not contact db_1 again in the second stage. In this case, the metasearcher only contacts db_2 in the second stage to retrieve its top ranking documents.

  Let the retrieval cost for an unprobed database be d and the cost for a probed database be d' ($d' \le d$). Then the probing and retrieval cost in the overall metasearching process is

$$|DB^P| \cdot c + |DB^k - DB^P| \cdot d + |DB^k \cap DB^P| \cdot d'$$

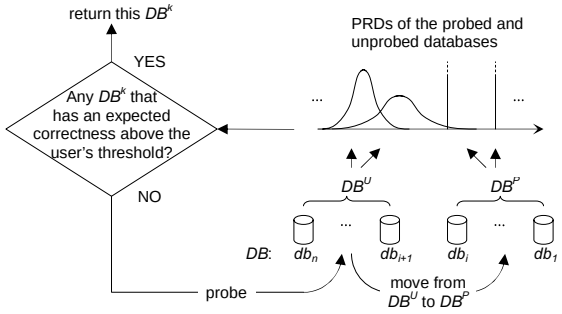


Figure 5: The dynamic probing process

In this paper, we mainly use the the probing cost model as our cost metric. Note that an optimal algorithm for the probing cost model may not be optimal for the PR cost model: Even if an algorithm does fewer probing during the first stage, the algorithm may incur a significant cost during the second stage if none of the returned databases was probed. However, the following theorem shows that under a certain condition, an optimal probing strategy for the probing cost model is also optimal for the PR cost model.

Theorem 3 *Under the condition that $DB^k \subseteq DB^P$ (i.e. all the returned databases have been probed), the optimal probing algorithm under the probing-only cost model is also optimal for the PR cost model.* □

Proof See Appendix ■

In our experiments, we observed that the condition in the above theorem is valid for most cases. That is, $DB^k \subseteq DB^P$ in the majority of cases, which means our algorithm is optimal also for the PR cost model.

3 The Dynamic Probing Algorithm

Given a query q, n databases and a threshold t, our goal is to use a minimum number of probing to find a k-subset DB^k whose expected correctness exceeds t. Figure 5 roughly illustrates our dynamic probing process to achieve this goal. At any intermediate step of the dynamic probing, the entire set of databases DB is divided into two subsets: the set of probed databases DB^P and the set of unprobed databases DB^U. Based on the impulse and regular PRDs of DB^P and DB^U, we compute the expected correctness of every k-subset DB^k (using Theorem 1 for the absolute correctness, for example). If there is a DB^k such that $E[Cor(DB^k)] \ge t$, the dynamic probing halts and returns this DB^k; otherwise it continues to probe one more database in

[4] Which will be necessary if our relevancy definition is document-similarity-based (Section 2.1)

Algorithm 3.1 $DPro(DB, q, k, t)$
Input:
DB: the entire set of given databases, $\{db_1,...,db_n\}$
q: a given query
k: the number of databases to return
t: the user's threshold for $E[Cor(DB^k)]$
Output:
DB^k with $E[Cor(DB^k)] \geq t$
Procedure
[1] $DB^P \leftarrow \emptyset$, $DB^U \leftarrow DB$
[2] If ($E[Cor(DB^k)] \geq t$) for some $DB^k \subseteq DB$
Return DB^k
[3] $db_i \leftarrow SelectDb(DB^U)$
[4] Probe db_i
[5] Change the PRD of db_i from regular to an impulse
[6] $DB^P \leftarrow DB^P \cup \{db_i\}$, $DB^U \leftarrow DB^U - \{db_i\}$
[7] Go to [2]

Figure 6: The dynamic probing algorithm $DPro$

DB^U, moves it from DB^U to DB^P, and recomputes the expected correctness for every DB^k.

Figure 6 provides the algorithm of our dynamic probing process. At each iteration, we try to find a k-subset DB^k that has the desired level of expected correctness and return it (Step [2]). If no such DB^k exists we pick a database from the unprobed set (Step [3]), probe it (Step [4]) and recompute the expected correctness (goto Step [2]).

Note that one key issue in this algorithm is how *SelectDb(DB^U)* should pick the next "best" database to probe in order to minimize probing cost. In the next subsection, we derive the answer to this question.

3.1 Selecting the optimal candidate database for probing

In *SelectDB(DB^U)*, we need to select the next database candidate that will lead to the earliest termination of the probing process and thus minimizing the probing cost. This database often should not be the one with the largest expected relevancy. Consider the following example.

Example 4 We want to return the top-2 databases from $\{db_1, db_2, db_3\}$. We have not probed any of them. Figure 7(a) shows their PRDs. We assume that $E[Cor(DB^2)]$ is smaller than the user threshold for any $DB^2 \subseteq \{db_1, db_2, db_3\}$ yet. We need to pick the next database to probe.

Note that we do not need to probe db_1, because its relevancy is the highest among all three, and it will always be returned as part of DB^2. Probing db^1 does not increase answer correctness at all. Similarly, note that probing db_2 is not very helpful, either. Because $r(db_2, q)$ lies between the two peaks of $r(db_3, q)$, even after we probe db_2 (Figure 7(b)), it is still uncertain which one (between db_2 and db_3) will have higher relevancy.

In constrast, probing db_3 is very likely to improve the certainty of our answer. Given the PRD of db_3, $r(db_3, q)$ will be either on the left side of $r(db_2, q)$ (Figure 7(c)) or on the right side (Figure 7(d)). If it is on the left side (Figure 7(c)), we can return $\{db_1, db_2\}$ as the top-2 databases. If it is on the right side (Figure 7(d)), we can return $\{db_1, db_3\}$ as the top-2 databases. In either case, we can return the top-2 databases with high confidence.

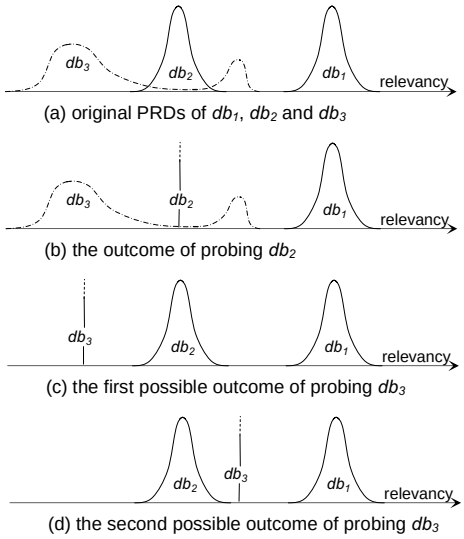


Figure 7: Selecting the top-2 databases from $\{db_1, db_2, db_3\}$

Algorithm 3.2 *SelectDb(DB^U)*
Input:
DB^U: the set of unprobed databases
Output:
db_i: the next database to probe
Procedure
[1] For every $db_i \in DB^U$:
[2] $cost_i = c + \mathsf{ECost}(DB^U - \{db_i\})$
[3] Return db_i with smallest $cost_i$

Figure 8: The optimal *SelectDb(DB^U)* function

Therefore, *SelectDb(DB^U)* should pick db_3 as the next database to probe, because we can finish the probing process only after one probing. Otherwise our algorithm needs at least two probing to halt. Notice that the expected relevancy of db_3 is the lowest among the three databases. The next database to probe is not the one with the highest expected relevancy. □

From this example, we can see that the function *SelectDb(DB^U)* should pick the $db_i \in DB^U$ that yields the *smallest number of expected probing*. To formalize this idea, we introduce the notation $\mathsf{ECost}(DB^U)$ to represent the expected amount of *additional* probing on DB^U after we have probed $DB - DB^U$ (=DB^P).

Now we analyze the expected probing cost if we pick $db_i \in DB^U$ as the next database to probe. The cost for probing db_i itself is c. The expected cost after probing db_i is $\mathsf{ECost}(DB^U - \{db_i\})$ under our notation. Therefore, by probing db_i next, we are expected to incur $c + \mathsf{ECost}(DB^U - \{db_i\})$ additional probing cost. Based on this understanding, we now describe the function *SelectDb(DB^U)* in Figure 8. In Steps [1] and [2], the algorithm first computes the expected additional probing cost for every $db_i \in DB^U$. Then Step [3] returns the one with the smallest cost.

Algorithm 3.3 $\mathit{ECost}(DB^U)$
Input:
DB^U: the set of unprobed databases
Output:
cost: the expected probing cost for DB^U
Procedure
[1] If ($E[Cor(DB^k)] \geq t$) for some $DB^k \subseteq DB$
Return 0
[2] For every $db_i \in DB^U$:
[3] $cost_i = c + \mathit{ECost}(DB^U - \{db_i\})$
[4] Return $\min(cost_i)$

Figure 9: Algorithm $\mathit{ECost}(DB^U)$

We now explain how we can compute $\mathit{ECost}(DB^U)$ using recursion. We assume that we have probed DB^P so far, and DB^U ($= DB - DB^P$) have not been probed yet. There are two possible scenarios at this point:

- *Case 1 (Stopping condition)*: With the databases DB^P probed, we can find a $DB^k \subseteq DB$ such that $E[Cor(DB^k)] \geq t$. In this case, we can simply return DB^k as the top-k databases. We do not need any further probing. Thus,

$$\mathit{ECost}(DB^U) = 0 \tag{4}$$

  Note that when all databases have been probed ($DB^U = \emptyset$), we know the exact relevancy of all databases, so $\mathit{ECost}(DB^U) = 0$.

- *Case 2 (Recursion)*: There is no $DB^k \subseteq DB$ whose expected correctness exceeds t. Therefore, we need to probe more databases to improve the expected correctness. Assume we probe $db_i \in DB^U$ next. Then the expected probing cost is $c + \mathit{ECost}(DB^U - \{db_i\})$. Remember that *SelectDb(DB^U)* always picks the db_i with the minimal expected cost. Therefore, the expected cost at this point is

$$\mathit{ECost}(DB^U) = \min_{db_i \in DB^U}(c + \mathit{ECost}(DB^U - \{db_i\})) \tag{5}$$

Figure 9 shows the algorithm to compute $\mathit{ECost}(DB^U)$. In Step [1], we first check whether we have reached the stopping condition. If not, we compute the expected probing cost for every $db_i \in DB^U$ (Steps [2] and [3]), and return the minimum expected cost (Step [4]).

The following theorem shows the optimality of our algorithm $SelectDb(DB^U)$.

Theorem 4 *$SelectDb(DB^U)$ returns the database that leads to the minimum expected probing cost, $\mathit{ECost}(DB^U)$, on the set of unprobed databases DB^U.* □

Proof See Appendix ■

Note that the computation of $\mathit{ECost}(DB^U)$ is recursive and can be very expensive. For example, assume that $DB^U = \{db_1, ..., db_n\}$ as we show in Figure 10. To compute $\mathit{ECost}(DB^U)$, we have to compute $\mathit{ECost}(DB^U - \{db_i\})$ for every $1 \leq i \leq n$ (first-level branching in Figure 10).

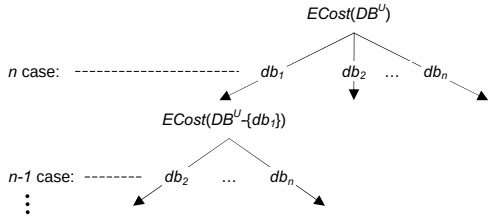


Figure 10: Exploring a search tree to compute $\mathit{ECost}(DB^U)$

Algorithm 3.4 *greedySelectDb(DB^U)*
Input:
DB^U: the set of unprobed databases
Output:
db_i: the next database to probe
Procedure
[1] For every $db_i \in DB^U$:
[2] $ECor_i = \max_{DB^k \subseteq DB}(E[Cor(DB^k)]$ after probing $db_i)$
[3] Return db_i with the highest $ECor_i$

Figure 11: The greedy *SelectDb(DB^U)* function

Then to compute $\mathit{ECost}(DB^U - \{db_i\})$, we need to compute $\mathit{ECost}(DB^U - \{db_i, db_j\})$ for every $j \neq i$ (second-level branching in Figure 10). Therefore, the cost for computing $\mathit{ECost}(DB^U)$ is $O(n!)$ if $|DB^U| = n$. Clearly this is too expensive when we mediate a large number of databases. In the next subsection, we propose a greedy algorithm that reduces the computation complexity of selecting the next database to $O(n)$.

3.2 A greedy choice

The goal of the *DPro* algorithm is to find a DB^k with $E[Cor(DB^k)] \geq t$ using minimum number of probing. Thus, the optimal *DPro* computes the expected probing cost for all possible probing scenarios and picks the one with the minimum cost. Informally, we may consider that the optimal *DPro* "looks all steps ahead" and picks the best one. Our new greedy algorithm, instead, looks only "one step ahead" and picks the best one.

The basic idea of our greedy algorithm is the following: Since we can finish our probing process when $E[Cor(DB^k)]$ exceeds t for some DB^k, the next database that we probe should be the one that leads to the highest $E[Cor(DB^k)]$ after probing (thus mostly likely to exceed t early).

Notice the subtle difference between the optimal algorithm and the greedy algorithm. The optimal algorithm computes $\mathit{ECost}(DB^U)$ for *all* possible scenarios, while our greedy algorithm computes $E[Cor(DB^k)]$ after we probe only *one* more database db_i. Using Theorem 1 we can compute $E[Cor_a(DB^k)]$ after we probe db_i if we know the PRD of each database.[5]

In Figure 11, we show a new *SelectDb(DB^U)* function that implements this greedy idea. In Steps [1] and [2], the algorithm computes the expected correctness value after we probe db_i. Then in Step [3] it returns the db_i that leads to the highest expected correctness.

[5]Since we do not know the outcome of probing db_i, we need to use Theorem 1 to compute and expected $E[Cor(DB^k)]$ value, based on db_i's PRD. The detailed formula is provided in the appendix.

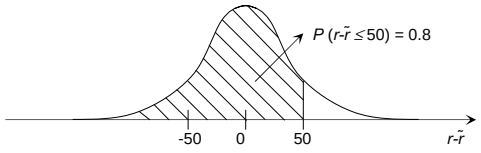


Figure 12: Distribution of the absolute-error function

4 Probabilistic Relevancy Distribution

In Section 2, we assumed that the PRD for database db was already given. We now discuss how we obtain the PRD that gives us $P(r(db, q) \leq \alpha \mid \tilde{r}(db, q) = \beta)$, where α and β are specific relevancy values. For simplicity, we use r and $\tilde{r}$ to represent $r(db, q)$ and $\tilde{r}(db, q)$.

Our basic idea is to use *sampling* to estimate the PRD. That is, we issue a small number of sampling queries, say 1000, to db and observe how the actual r values are distributed. From this result, we can compute the difference of r from $\tilde{r}$ and obtain the distribution.

Note that the PRD $P(r \leq \alpha \mid \tilde{r} = \beta)$ is conditional on $\tilde{r}$. Therefore, the exact shape of the PRD may be very different for different $\tilde{r}$ values. Ideally, we have to issue a number of sampling queries for each $\tilde{r}$ value, in order to obtain the correct PRD shape for each $\tilde{r}$. However, issuing a set of queries for each $\tilde{r}$ is too expensive given that there are an infinite number of $\tilde{r}$ values. To reduce the cost for PRD estimation, we assume that the distribution we observe is independent of what $\tilde{r}$ may be. More precisely, we may consider one of the following independence assumptions:

- *Absolute-error independence*: We assume that the absolute error of our estimate, $r - \tilde{r}$ (the difference between our estimate and the actual relevancy), is independent of the $\tilde{r}$ value. Therefore, from our sampling queries, we obtain a single distribution for $(r - \tilde{r})$ values (even if the $\tilde{r}$ values for the queries are different), and use the distribution to derive the PRD.
- *Relative-error independence*: We assume that the relative error of our estimation, $\frac{r-\tilde{r}}{\tilde{r}}$, is independent of the $\tilde{r}$ value. Therefore, from sampling queries, we obtain a single distribution for $\frac{r-\tilde{r}}{\tilde{r}}$ values (even if their $\tilde{r}$ values for the queries are different) and use the distribution to derive the PRD.

In general, if there is an error function $err(r, \tilde{r})$ (e.g., $err(r, \tilde{r}) = r - \tilde{r}$ for the first case) whose distribution is independent of $\tilde{r}$, then we can use just one set of queries (regardless of their $\tilde{r}$ values) to estimate the $err(r, \tilde{r})$ distribution. Then using this distribution, we can obtain the correct PRD for every $\tilde{r}$ value. This can be illustrated through the following example:

Example 5 Suppose from 1,000 sampling queries, we are able to obtain a probability distribution for the absolute-error function: $err(r, \tilde{r}) = r - \tilde{r}$, as shown in Figure 12. Assume that $r - \tilde{r}$ is independent of $\tilde{r}$. Let us derive the probability $P(r \leq 150 \mid \tilde{r} = 100)$ using this distribution.

$$
\begin{aligned}
&P(r \leq 150 \mid \tilde{r} = 100) \\
&= P(r - \tilde{r} \leq 50 \mid \tilde{r} = 100) \\
&= P(r - \tilde{r} \leq 50) \qquad \text{(independency of } r - \tilde{r} \text{ and } \tilde{r}\text{)}
\end{aligned}
$$

This probability $P(r - \tilde{r} \leq 50)$, as shown in Figure 12, is 0.8. □

More formally, we observe that the error function $err(r, \tilde{r})$ should satisfy the following properties to derive a PRD:

I. **Independency**:
$err(r, \tilde{r})$ is probabilistically independent of $\tilde{r}$

II. **Monotonicity**:
$err(r_1, \tilde{r}) \leq err(r_2, \tilde{r})$ for any $r_1 \leq r_2$

The following theorem shows that the probability of a relevancy value can be obtained through the probability of the error function, via a variable transformation from r to $err(r, \tilde{r})$.

Theorem 5 *If* $err(r, \tilde{r})$ *is* independent *and* monotonic, *then*

$$P(r \leq \alpha \mid \tilde{r} = \beta) = P(err(r, \tilde{r}) \leq err(\alpha, \beta)) \quad (6)$$

□

Proof See Appendix. ■

In Section 5, we compare the absolute-error function $err_a(r, \tilde{r}) = r - \tilde{r}$, and the relative-error function, $err_r(r, \tilde{r}) = \frac{(r-\tilde{r})}{\tilde{r}}$ experimentally. Our result shows that the relative-error function works well in practice and roughly satisfies the two properties in Theorem 5.

5 Experiments

This section reports our experimental results that testify the effectiveness of the dynamic probing approach. Section 5.1 describes the experimental setup and the dataset we use. Section 5.2 experimentally compares the error functions to derive a PRD. Sections 5.3 and 5.4 show the improvement of our dynamic probing compared to the existing methods.

5.1 Experimental setup

In our experiments, we simulate a real metasearching application by mediating 20 *real* Hidden-Web databases and using 4,000 *real* Web query traces. The databases for our experiments are mainly related to "health." Thus, we may consider that the experiments evaluate the effectiveness of our dynamic probing approach in the context of a health-related metasearcher. In this subsection, we explain our experimental setup in detail.

First, we select 13 databases from the health category of InvisibleWeb,[6] which is a manually-maintained directory of Hidden-Web databases. While the directory lists more than 13 health-related databases, most of them are either obsolete or too small. In our experiments, we use only the databases with at least 3,000 documents. Most of the small databases are relatively obscure and of little interest.

Because 13 is a relatively small number, and in order to introduce heterogeneity to our experiments, we append four more databases on broader topics (e.g., Science and Nature), and three more news websites (e.g., CNN and NYTimes). We show some sample databases and their sizes in Figure 13.[7] The complete list of our databases can be found in [20].

Second, we select a subset of queries from a real query trace from Yahoo (provided by Overture[8]). We start by building a sample medical vocabulary using single terms extracted from the health topic pages in MedLinePlus,[9] an authoritative medical information website. We then randomly pick any 2-term and 3-term queries from the Yahoo query trace that use at least two terms from our vocabulary. Again, this selection was done to simulate a metasearcher that focuses on health-related topics.

[6] `http://www.invisibleweb.com`

[7] For databases that do not export their sizes, we roughly estimate the size by issuing a query with common terms, e.g. "medical OR health OR cancer ..."

[8] `http://inventory.overture.com/`

[9] `http://www.medlineplus.org`

Database	URL	Size
MedWeb	www.medweb.emory.edu	∼14,000
PubMed Central	www.pubmedcentral.nih.gov	∼60,000
NIH	www.nih.gov	163,799
Science	www.sciencemag.org	29,652

Figure 13: Sample Web databases used in our experiment

Using the above selection method, we prepare a sample query set QS_1 which contains 1,000 2-term queries and 1,000 3-term queries. We use QS_1 in Section 5.2 to derive the PRD for each database. Similarly, we prepare another query set QS_2 that contains, again, 1,000 2-term queries and 1,000 3-term queries. QS_2 is used in Sections 5.3 through 5.4 when we evaluate how well our dynamic prober works. Note that a typical Web query has only a small number of terms, with 2.2 terms on average [19]. Therefore, we believe our experiments using 2 or 3-term queries reflect the typical scenario that a real metasearcher can expect.

In all of our experiments, we use *document-frequency-based* relevancy metric (Section 2.1) and independence relevancy estimator (Eq.1). Further, we use the independency estimator to create a baseline representing traditional estimation-based selection methods. All of our dynamic probing is done using the greedy algorithm (Section 3.2). Due to its exponential computational cost, it takes a long time for the optimal algorithm to terminiate, so we could not finish enough experiments to include their results in this draft.

5.2 Selecting an error function to derive the correct PRD

To obtain the correct PRD from the $err(r, \tilde{r})$ distribution, $err(r, \tilde{r})$ needs to be *monotonic* and *independent* of $\tilde{r}$ (Theorem 5). In this subsection, we experimentally compare the absolute-error function $err_a(r, \tilde{r}) = r - \tilde{r}$ with the relative-error function, $err_r(r, \tilde{r}) = \frac{(r-\tilde{r})}{\tilde{r}}$, and select the better one for our experiment.

From their analytical forms, it is easy to verify that both error functions are monotonic. What we need to verify is the *independence* property. This can be done by computing the statistical correlation between $err(r, \tilde{r})$ and $\tilde{r}$, where err can be err_a or err_r. If the correlation is close to 0, it means err and $\tilde{r}$ are roughly independent; Otherwise they are not.

More specifically, we first obtain the $err(r, \tilde{r})$ value for every 1,000 2-term query in QS_1 on a database db. We then compute the correlation between $err(r, \tilde{r})$ and $\tilde{r}$ on these 1,000 queries, regarding db. We repeat this process for all 20 databases and compute the average correlation over all databases. We similarly compute the correlation for the 1,000 3-term sample queries in QS_1, and summarize the results in Figure 14. The max correlation value among the 20 databases is also included to show the extreme cases. Figure 14(a) shows that the absolute error $err_a(r, \tilde{r})$ has a high positive correlation with $\tilde{r}$ for both 2-term and 3-term queries. Therefore, $err_a(r, \tilde{r})$ is dependent on $\tilde{r}$ and becomes larger as $\tilde{r}$ gets larger. Figure 14(b) reveals that the relative error $err_r(r, \tilde{r})$ is roughly independent of $\tilde{r}$. Therefore we use $err_r(r, \tilde{r})$ as the error function to derive a PRD.

From our experiments, we observe that the shape of $err_r(r, \tilde{r})$ distribution for 2-term queries is slightly different from that for 3-term queries. Therefore, we maintain two PRDs for each database, one for 2-term queries and the other for 3-term queries, and pick the appropriate PRD depending on the number of terms in a query.

Correlation between err_a and $\tilde{r}$	2-term	3-term
average	0.567	0.632
max	0.851	0.885

(a) The average and maximum correlation between $err_a(r, \tilde{r})$ and $\tilde{r}$ for 20 databases

Correlation between err_r and $\tilde{r}$	2-term	3-term
average	-0.073	-0.023
max	0.217	0.061

(b) The average and maximum correlation between $err_r(r, \tilde{r})$ and $\tilde{r}$ for 20 databases

Figure 14: The correlation between $err(r, \tilde{r})$ and $\tilde{r}$

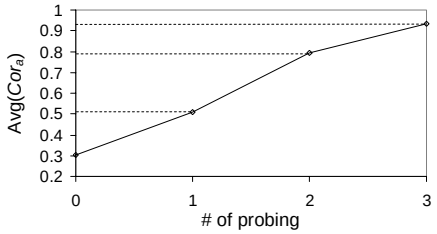


Figure 15: The effect of dynamic probing on the average correctness ($k = 1$,$t = 0.9$)

5.3 Effectiveness of dynamic probing

In the second set of our experiments, we study the impact of dynamic probing on the correctness of database selection. Our main goal in this section is to investigate how accurate an answer becomes as we probe more databases, so we restrict our experiments only to the queries that require at least three probing for *DPro* to terminate (i.e. $E[Cor(DB^k)] \geq t$ only after three probing). When we set $t = 0.9$ and $k = 1$ [10] as our parameters, 1,033 out of the 2,000 test queries in QS_2 (1,000 2-term queries and 1,000 3-term queries) belong to this category.

For each query issued, we then ask $DPro$ to report the database with the highest expected correctness after *each* probing (even if it has not terminated yet). By comparing this reported database to the most relevant database (i.e., $DB^k = DB^{topk}$?) we can measure how accurate the answer becomes as we probe more databases. Note the correct DB^{topk} is inaccessible to *DPro* during its probing process.

Figure 15 summarizes the result from these experiments. In the figure, the horizontal axis shows the number of probing that *DPro* has performed so far. The vertical axis shows the fraction of correct answers that $DPro$ reports at the given number of probing. For example, after one probing, *DPro* reports the correct database for 524 queries out of 1,033, so the average correctness is $524/1,033 = 0.51$ at one probing.

Note that at the point of no probing (# of probing = 0), *DPro* is identical to the traditional estimation-based method because it does not use any dynamic probing. At this point average correctness is only 0.30. After two probing correctness reaches 0.80. From this result, it is clear that dynamic probing significantly improves the answer correctness: We can improve the correctness of the answer by more than twice with only two probing.

[10] Note that Cor_a and Cor_p are the same when $k = 1$. Therefore we do not specify our correctness metric in this experiment.

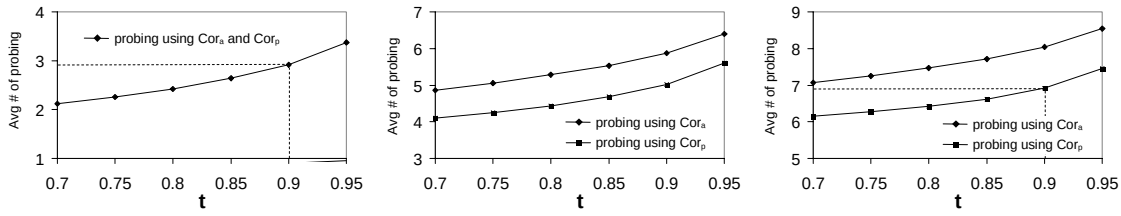


Figure 16: The average number of probing under different settings of t and k

5.4 The average amount of probing under different t settings

In this subsection, we study how many probings *DPro* does for different settings of t. We experiment on six t values: $\{0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$. For a larger t, it is expected that $DPro$ probes more databases to meet the threshold. In Figure 16, we show how the number of probing increases as t becomes larger. The x-axis shows the different t values, and the y-axis is the average number of probing $DPro$ does for a particular t, over the 2,000 test queries in QS_2. For example, when $k = 1$ (Figure 16(a)), *DPro* terminates after 3 probing on average for the threshold value $t = 0.9$. In Figures 16(b) and (c) we include the results for the absolute (Cor_a) and partial (Cor_p) correctness metrics. When $k = 1$, the two correctness metric are the same, so we have only one graph in Figure 16(a). Note that the graph for Cor_a is always above that of Cor_p. Since Cor_p is always larger than Cor_a, *DPro* reaches the correctness threshold faster under Cor_p and terminates earlier.

The figure shows that our algorithm *DPro* can find correct databases with a reasonable number of probing. For example, when $k = 5$ and $t = 0.9$ (Figure 16(c)), *DPro* finds a DB^k with $E[Cor_p(DB^k)] > 0.9$ after 6.8 probing. In most cases, all 5 returned databases are probed during the selection process. That means that 5 of 6.8 probing is done on the top-k databases returned, so the information that we collect during the probing stage can be used to reduce the cost for the document retrieval stage (Figure 4). So the "extra" probing in the overall metasearching process is only 1.8.

Note that even if the user specified threshold t is 0.7, the top-k databases that *DPro* returns may be correct in more than 70% of the time. User threshold t is simply a lower bound for the correctness of the returned answer. To show how accurate answers *DPro* returns, Figure 17 and Figure 18 show the average correctness of the answers for different threshold values. Figure 17 shows the result under the Cor_a metric, and Figure 18 shows the result under the Cor_p metric. The baseline (the triangle line) is the average correctness of the traditional estimation-based selection. Since the traditional method does not depend on the t value, the average correctness remains constant. The dotted lines in the figures represent $Avg(Cor) = t$. The average correctness of the answers from *DPro* should be higher than the dotted line, since t is the minimum threshold value for *DPro* to terminate. From the graphs, we can see that this is indeed the case.

6 Related work

Database selection is a critical step in the metasearching process. Past research mainly focused on applying certain approximate method to estimate how relevant a database is to the user's query. The databases with the highest estimated relevancy are selected and presented to the user. The quality of database selection is highly dependent on the accuracy of the estimation method. In the early work of bGlOSS [14] that mediates databases with boolean search interfaces, a metasearcher estimates the relevancy of each database by assuming query terms appear independently. vGlOSS [15] extends bGloss to support databases with vector-based search interfaces, and uses a *high-correlation assumption* or a *disjoint assumption* on query terms to estimate the relevancy of a database under the vector-space-model. [21] uses term covariance information to model the dependency between each pair of terms, and achieve better estimation than vGlOSS. An even better estimation is reported in [25] by incorporating document linkage information. There have been parallel research in the distributed information retrieval context. In [2, 5, 24] the relevancy of a database is modelled by the probability of the database containing similar documents to the query. In [4], various estimation methods discussed above are compared on a common basis. Our dynamic probing method is orthogonal to these research in that we are not proposing a new estimation method under certain relevancy definition. Instead, we use probabilistic distribution to model the accuracy of a particular estimation method, and use probing to increase the correctness of database selection.

Database selection is related to a broader research area called *top-k query answering*. Past research [11, 7, 8, 9] largely focused on relational data, and use deterministic methods to find the absolutely correct top-k answers. While in our context of Hidden-Web-database-selection, enforcing the deterministic approach would end up probing almost all the Hidden-Web databases. In our probabilistic approach, we only probe the databases that would maximally increase our certainty of the top-k answers.

Mediating heterogenous databases to provide a single query interface has been studied for years [17, 12]. While the existing research focused on integrating data sources with relational search capabilities, we in this paper investigate the mediation of Hidden-Web databases with much more primitive query interfaces over a collection of unstructured textual data.

7 Conclusion

We have presented a new approach to the Hidden Wed database selection problem using dynamic probing. In our approach, the accuracy of a particular relevancy estimator is modelled using Probabilistic Relevancy Distribution (PRD). The PRD enables us to quantify the correctness of a particular top-k answer set in a probabilistic sense. We propose an optimal probing strategy that uses the least probing to reach the user-specified correctness threshold. A greedy probing strategy with much less computation complexity is also presented. Our experimental results reveal that dynamic probing significantly improves the answer's correctness with a reasonably small amount of probing.

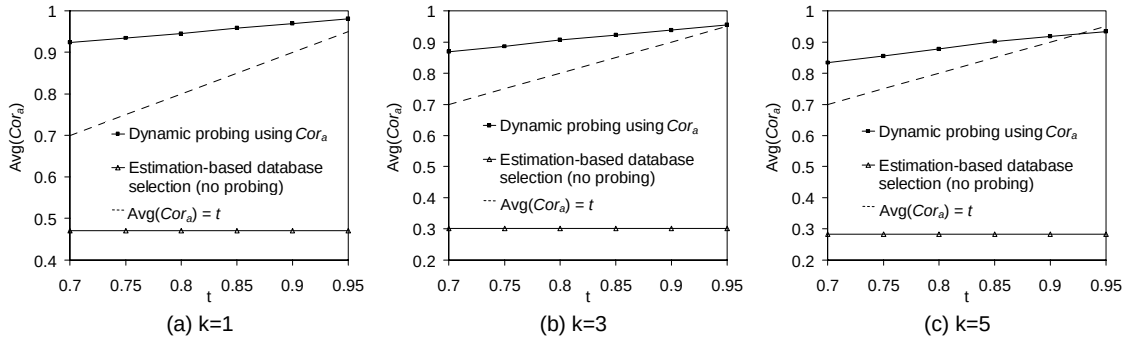


Figure 17: $Avg(Cor_a)$: dynamic probing vs. the estimation-based database selection

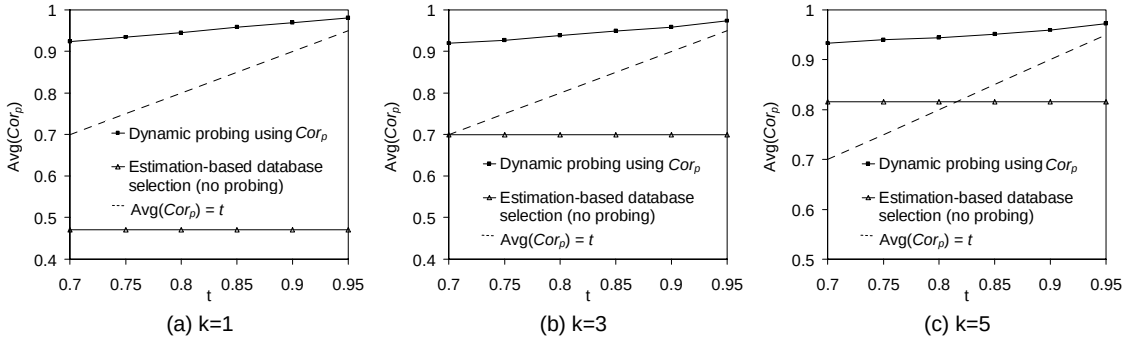


Figure 18: $Avg(Cor_p)$: dynamic probing vs. the estimation-based database selection

Our experimentation on real datasets justifies an effective new direction for the metasearching research. In the past, researchers tried to improve the correctness of database selection via constructing more accurate estimators that estimates the database's relevancy to a particular query. A more accurate estimator demands more comprehensive content-summary of each database. For example, storing the pair-wise term covariance [21, 25] takes $O(M^2)$ of space, where M is the size of the vocabulary. However, once the estimator is constructed, the correctness of database selection is fixed at a certain level and cannot be explicitly controlled by the user. In our dynamic probing approach, the user explicitly specifies the desired level of correctness, regardless of what estimator we use. Our results reveal that using the relevancy estimator developed in bGlOSS [14], the answer's correctness is greatly improved via a small amount of probing.

References

[1] M.K. Bergman. The Deep Web: Surfacing Hidden Value. Accessible at `www.brightplanet.com/deepcontent/tutorials/DeepWeb`, 2000

[2] C. Baumgarten. A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval. In *Proc. of ACM SIGIR '99*, CA, 1999

[3] J.A. Borges, I. Morales, N.J. Rodrguez. Guidelines for Designing Usable World Wide Web Pages. In *Proc. of ACM SIGCHI '96*, `http://sigchi.org/chi96/`, 1996

[4] Craswell, P. Bailey, and D. Hawking. Server Selection on the World Wide Web. In *Proc. of ACM Conf. Digital Library '00*, TX, 2000

[5] J. P. Callan, Z. Lu, and W. Croft. Searching Distributed Collections with Inference Networks. In *Proc. of ACM SIGIR '95*, WA, 1995

[6] J. Callan, M. Connell, and A. Du. Automatic Discovery of Language Models for Text Databases. In *Proc. of ACM SIGMOD '99*, PA, 1999

[7] S. Chaudhuri and L. Gravano. Optimizing Queries over Multimedia Repositories. In *Proc. of ACM SIGMOD '96*, Canada, 1996

[8] S. Chaudhuri and L. Gravano. Evaluating Top-k Selection Queries. In *Proc. of VLDB '99*, Scotland, 1999

[9] K. Chang and S.Hwang. Minimal Probing: Supporting Expensive Predicates for Top-k Queries. In *Proc. of ACM SIGMOD '02*, WI, 2002

[10] A. Clyde. The Invisible Web. *Teacher Librarian* 29(4), `http://www.teacherlibrarian.com`, 2002

[11] R. Fagin. Combining fuzzy information from multiple systems. In *Proc. of ACM PODS '96*, Canada, 1996

[12] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, et al. The TSIMMIS Project: Integration of Heterogeneous Information Sources. *J. Intelligent Information System* 8(2):117-132, 1997

[13] M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1990

[14] L. Gravano, H. Garcia-Molina, A. Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. In *Proc. of SIGMOD 94*, MN, 1994

[15] L. Gravano, and H. Garcia-Molina. Generalizing GlOSS to Vector-Space databases and Broker Hierarchies. In *Proc. of VLDB '95*, Switzerland, 1995

[16] L. Gravano, H. Garcia-Molina, A. Tomasic. GlOSS: Text-Source Discovery over the Internet. *ACM TODS* 24(2):229-264, 1999

[17] A.Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal* 10(4):270-294, 2001.

[18] P.G. Ipeirotis, L. Gravano. Distributed Search over the Hidden Web: Hierarchical Database Sampling and Selection. In *Proc' of VLDB 02*, China, 2002

[19] S. Kirsch. The Future of Internet Search: Infoseek's Experiences Searching the Internect. *ACM SIGIR Forum* 32(2):3-7, 1998

[20] V.Z. Liu, R.C. Luo, J. Cho, W.W. Chu. A Probabilistic Framework for Hidden Web Database Selection Using Dynamic Probing. Technical report, UCLA, Computer Science Department, 2003

[21] W. Meng, K.L. Liu, C. Yu, et al. Determining Text Databases to Search in the Internet. In *Proc. of VLDB 98*, NY, 1998

[22] G. Salton, M.J. McGill. *Introduction to Modern Information Retrieval*, 1983

[23] K. Wiseman. The Invisible Web: Searching the Hidden Parts of the Web. *Learning Technology Review*, Fall 1999 / Winter 2000, `http://www.apple.com/education/LTReview/fall99`, 1999

[24] J. Xu and J. Callan. Effective Retrieval with Distributed Collections. In *Proc. of ACM SIGIR '98*, Australia, 1998

[25] C. Yu, W. Meng, W. Wu, K. Liu. Effi cient and Effective Metasearch for Text Databases Incorporating Linkages among Documents. In *Proc. of SIGMOD 01*, CA, 2001.

[26] B. Yuwono, D.L. Lee. Server Ranking for Distributed Text Retrieval System on the Internet. In *Proc. of the 5th Int'l Conf. on Database System for Advanced Applications*, Australia, 1997

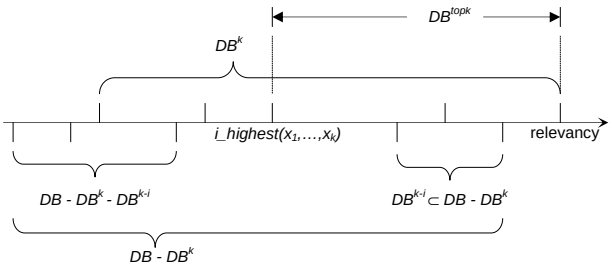


Figure 1: The event that $|DB^k \cap DB^{topk}| = i$

A The expected absolute correctness

We label the databases in DB^k as db_1, db_2, ..., db_k, and label the databases in $DB - DB^k$ as db_{k+1}, ..., db_n. Let $f_j(x_j)$ be the Probability Density Function derived from db_j's PRD ($1 \leq j \leq n$), and x_j be one possible value of $r(db_j, q)$.

Theorem 1 *Assuming that all databases operate independently,*

$$E[Cor_a(DB^k)] = \int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} \left(\prod_{db \in DB - DB^k} P(r(db, q) < \min(x_1, ..., x_k)) \right) \times \left(\prod_{1 \leq j \leq k} f_j(x_j) \right) dx_1 ... dx_k \quad (1)$$

where $\min(x_1, ..., x_k)$ *is the minimum relevancy value among all the* $db_j \in DB^k$. □

Proof From our definition of the expected absolute correctness,

$$E[Cor_a(DB^k)] = P(|DB^k \cap DB^{topk}| = k)$$

Now we need to derive the formula for $P(|DB^k \cap DB^{topk}| = k)$, the probability of DB^k equals to DB^{topk}.

Let us first assume that we know the relevancy values of all the databases in DB^k. Specifically, let x_j be the relevancy value taken by database db_j. $\min(x_1, ..., x_k)$ computes the minimum relevancy value among all $db_j \in DB^k$. Thus, the event that DB^k equals to DB^{topk}, i.e. DB^k is absolutely correct, is the same as the event that all the databases not in DB^k have relevancy lower than $\min(x_1, ..., x_k)$. Assuming that all the database operate independently, the probability of the latter event is:

$$\prod_{db \in DB - DB^k} P(r(db, q) < \min(x_1, ..., x_k)) \quad (2)$$

Since we may not know the exact relevancy values of databases in DB^k, we have to integrate Eq.(2) over all possible values of x_j for each $db_j \in DB^k$. This gives us the formula in Eq.(1). ■

B The expected partial correctness

Theorem 2 *Assuming that all databases operate independently,*

$$E[Cor_p(DB^k)] = \sum_{1 \leq i \leq k} \frac{i}{k} \cdot \int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} \left(\sum_{\substack{DB^{k-i} \subseteq \\ DB - DB^k}} \prod_{db \in DB^{k-i}} P(r(db, q) > i\_highest(x_1, ..., x_k)) \cdot \prod_{\substack{db' \in (DB - \\ DB^k - DB^{k-i})}} P(r(db', q) < i\_highest(x_1, ..., x_k)) \right) \times \left(\prod_{1 \leq j \leq k} f_j(x_j) \right) dx_1 ... dx_k \quad (3)$$

where $i\_highest(x_1, ..., x_k)$ *is a function that computes the* i_{th} *highest relevancy value among all the* $db_j \in DB^k$. □

Proof From out definition of the expected partial correctness,

$$E[Cor_p(DB^k)] = \sum_{1 \leq i \leq k} \frac{i}{k} \cdot P(|DB^k \cap DB^{topk}| = i) \quad (4)$$

Clearly, the critical part of computing Eq.(4) is to compute the probability $P(|DB^k \cap DB^{topk}| = i)$, $1 \leq i \leq k$. Similar to the proof for Theorem 1, we first assume that we know the relevancy values of all the databases in DB^k, with db_j's relevancy being x_j ($1 \leq j \leq k$). Then let $i\_highest(x_1, ..., x_k)$ represent the i_{th} highest relevancy value for all the databases in DB^k. The event that DB^k and DB^{topk} have i members in common is the same as that the i databases in DB^k with the highest relevancy values are also in DB^{topk}, while the rest $k - i$ databases in DB^k are not in DB^{topk}. Therefore, DB^{topk} must consist of two parts: first, the i database with the highest relevancy values in DB^k; second, DB^{k-i}, a $(k-i)$-subset of $DB - DB^k$. This is illustrated by Figure 1. All the databases in DB^{k-i} have relevancy values higher than $i\_highest(x_1, ..., x_k)$, while all the databases in $DB - DB^k - DB^{k-i}$ have relevancy values lower than $i\_highest(x_1, ..., x_k)$. Thus, by assuming all databases operate independently, the latter event has the following probability:

$$\prod_{db \in DB^{k-i}} P(r(db, q) > i\_highest(x_1, ..., x_k)) \cdot \prod_{\substack{db' \in (DB - \\ DB^k - DB^{k-i})}} P(r(db', q) < i\_highest(x_1, ..., x_k))$$

Since DB^{k-i} can be any $(k-i)$-subset of $DB - DB^k$, we need to enumerate over all possible $(k-i)$-subsets of $DB - DB^k$, and also integrate over all possible relevancy values taken by the databases in DB^k, which gives us Eq.(3). ■

C Invariance of the optimal probing algorithm under different cost models

Theorem 3 *Under the condition that $DB^k \subseteq DB^P$ (i.e. all the returned databases have been probed), the optimal probing algorithm under the probing-only cost model is also optimal for the PR cost model.* □

Proof Let algorithm A_1 be the optimal under the probing-only cost model, i.e. A_1 spends the minimum number of probing under the the probing-only model. For a particular query q, A_1 returns DB_1^k as a subset of DB_1^P, where DB_1^P is the set of databases A_1 probes. The total cost of A_1 under the total cost model is:

$$|DB_1^P| \cdot c + |DB_1^k - DB_1^P| \cdot d + |DB_1^k \cap DB_1^P| \cdot d'$$

which can be reduced to

$$|DB_1^P| \cdot c + |DB_1^k| \cdot d' = |DB_1^P| \cdot c + k \cdot d' \quad (5)$$

because $DB_1^k - DB_1^P = \emptyset$.

Now we want to show that A_1 is also optimal for q under the total cost model. We shall prove this via contradiction. Suppose for q, $A_2 \neq A_1$ is the optimal algorithm for q under the total cost model. Suppose A_2 find the satisfactory answer DB_2^k via probing DB_2^P. The total cost of A_2 under the total cost model is:

$$|DB_2^P| \cdot c + |DB_2^k - DB_2^P| \cdot d + |DB_2^k \cap DB_2^P| \cdot d' \quad (6)$$

Because $d \geq d'$, we have:

$$\begin{aligned} &|DB_2^P| \cdot c + |DB_2^k - DB_2^P| \cdot d + |DB_2^k \cap DB_2^P| \cdot d' \\ &\geq |DB_2^P| \cdot c + |DB_2^k - DB_2^P| \cdot d' + |DB_2^k \cap DB_2^P| \cdot d' \\ &= |DB_2^P| \cdot c + |DB_2^k| \cdot d' = |DB_2^P| \cdot c + k \cdot d' \end{aligned}$$

Since A_2 is optimal under the total cost model, we know Eq.(5) > Eq.(6). Therefore,

$$|DB_1^P| \cdot c + k \cdot d' > |DB_2^P| \cdot c + k \cdot d'$$

which gives us that $|DB_1^P| > |DB_2^P|$. Therefore, the cost of A_2 under the probing-only cost model is $|DB_2^P| \cdot c$ and is smaller than A_1's probing cost. This contradicts our assumption that A_1 is optimal under the probing-only cost model. ■

D Optimality of the $SelectDb(DB^U)$ function

Theorem 4 *$SelectDb(DB^U)$ returns the database that leads to the minimum expected probing cost, $\mathsf{ECost}(DB^U)$, on the set of unprobed databases DB^U.* □

Proof Based on Lemma 1 (which is given later), $\mathsf{ECost}(DB^U - \{db_i\})$ represents the minimum cost that we expect to spend, after we have probed db_i. Therefore, in Step [2] of Figure 8, $cost_i$ is the minimum expected cost incurred by probing db_i. By selecting the db_i that yields the smallest $cost_i$, we return the database that incurs the minimum expected cost over all possible probing scenarios. ■

Lemma 1 *$\mathsf{ECost}(DB^U)$ represents the minimum expected cost that we need to selectively probe DB^U.* □

Proof We shall prove by natural induction on the size of DB^U, i.e. $|DB^U|$.

1. $|DB^U| = 0$

   According to the stopping condition (Eq. (4) in Section 3.1), $\mathsf{ECost}(DB^U) = 0$. Since $\mathsf{ECost}(DB^U)$ is non-negative by definition, it is minimum in this case.

2. $|DB^U| > 0$. We can assume that for all $DB^{U'}$ s.t. $|DB^{U'}| = |DB^U| - 1$ (i.e. $DB^{U'}$ contains one less database than DB^U), $\mathsf{ECost}(DB^{U'})$ represents the minimum expected cost that we will spend on $DB^{U'}$. There are two sub-cases:

   (a) With all the databases probed DB^P ($= DB - DB^U$), we are able to find a $DB^k \subseteq DB$ s.t. $E[Cor(DB^k)] \geq t$ (t being the user-specified threshold). According to the stopping condition, $\mathsf{ECost}(DB^U) = 0$. $\mathsf{ECost}(DB^U)$ is minimum under this sub-case.

   (b) With all the databases probed DB^P, we cannot find a $DB^k \subseteq DB$ s.t. $E[Cor(DB^k)] \geq t$. According to the recursive definition:

   $$\begin{aligned} &\mathsf{ECost}(DB^U) \\ &= \min_{db \in DB^U} (c + \mathsf{ECost}(DB^U - \{db\})) \end{aligned} \quad (7)$$

   Since for every $db \in DB^U$, $\mathsf{ECost}(DB^U - \{db\})$ is the minimum expected cost we will spend on $DB^U - \{db\}$ (according to our induction assumption), Eq.(7) computes the minimum expected cost we will spend on DB^U. ■

E Computing the expected value of $E[Cor(DB^k)]$ for the greedy probing strategy

In Step[2] of Figure 11, $ECor_i$ is the maximum $E[Cor(DB^k)]$ value we can obtain after probing db_i, for a certain $DB^k \subseteq DB$. The key issue is how can we know the $E[Cor(DB^k)]$ value for any DB^k after probing db_i, when we have not yet probed db_i? Our solution is to compute an expected $E[Cor(DB^k)]$, based on the PRD of db_i.

Specifically, let us assume α is the relevancy of db_i after probing. Under this condition, we can use Theorem 1 to compute $E[Cor_a(DB^k) \mid r(db_i, q) = \alpha]$, which means the expected correctness of DB^k under the condition that $r(db_i, q) = \alpha$. We can then compute the expected $E[Cor(DB^k)]$ as the average over all possible α values that we can obtain through probing db_i:

$$\begin{aligned} &E[E[Cor(DB^k)]] = \\ &\int_{-\infty}^{+\infty} \left(\max_{DB^k \subseteq DB} E[Cor(DB^k) \mid r(db_i, q) = \alpha] \right) \cdot f_i(\alpha) \cdot d\alpha \end{aligned} \quad (8)$$

where $f_i(\alpha)$ is the probability density at α, which is given by db_i's PRD.

Formally, we can replace the "$E[Cor(DB^k)]$ if we probe db_i" in Step [2] of Figure 11 as $E[E[Cor(DB^k)]]$ computed by Eq. 8.

F Relationship between the PRD and the $err(r, \tilde{r})$ distribution

Theorem 5 *If* $err(r, \tilde{r})$ *is* independent *and* monotonic*, then*

$$P(r \leq \alpha \mid \tilde{r} = \beta) = P(err(r, \tilde{r}) \leq err(\alpha, \beta))$$

□

Proof

$$\begin{aligned}
&P(r \leq \alpha \mid \tilde{r} = \beta) \\
&= P(err(r, \tilde{r}) \leq err(\alpha, \tilde{r}) \mid \tilde{r} = \beta) && (Monotonicity) \\
&= P(err(r, \tilde{r}) \leq err(\alpha, \beta) \mid \tilde{r} = \beta) && (\tilde{r} = \beta) \\
&= P(err(r, \tilde{r}) \leq err(\alpha, \beta)) && (Independency)
\end{aligned}$$

■