Discovering and Ranking Data Intensive Web Services: A Source-Biased Approach

James Caverlee, Ling Liu, and Daniel Rocco
Georgia Institute of Technology
College of Computing
Atlanta, GA 30332, U.S.A.
{caverlee, lingliu, rockdj}@cc.gatech.edu

ABSTRACT

This paper presents a novel source-biased approach to automatically discover and rank relevant data intensive web services. It supports a service-centric view of the Web through source-biased probing and source-biased relevance detection and ranking metrics. Concretely, our approach is capable of answering source-centric queries by focusing on the nature and degree of the topical relevance of one service to others. This source-biased probing allows us to determine in very few interactions whether a target service is relevant to the source by probing the target with very precise probes and then ranking the relevant services discovered based on a set of metrics we define. Our metrics allow us to determine the nature and degree of the relevance of one service to another. We also introduce a performance enhancement to our basic approach called source-biased probing with focal terms. We also extend the basic probing framework to a more generalized service neighborhood graph model. We discuss the semantics of the neighborhood graph, how we may reason about the relationships among multiple services, and how we rank services based on the service neighborhood graph model. We also report initial experiments to show the effectiveness of our approach.

1. INTRODUCTION

The past few years have witnessed great strides in the accessibility and manageability of vast amounts of Web data. In particular, the widespread adoption of general purpose search engines like Google and AllTheWeb has added a layer of organization to an otherwise unwieldy medium. Typically, a search engine is optimized to identify a ranked list of Web pages relevant to a user query. This *page-centric* view of the Web has proven immensely successful.

But with the rise of high-quality data intensive web services on the so-called Deep Web (or Hidden Web) and the emergence of the web services paradigm, there is a growing demand for a new class of queries optimized not on the page level, but on the more general service level. Rather than requesting the top-ranked Web pages containing a certain keyword, say "autism", a user may be more interested in *service-centric* queries. For example, a user familiar with the popular online medical literature site PubMed may be interested in posing some of the following queries:

- What other data services are most similar to PubMed?
- Find other data services more general than PubMed? Or more specialized?
- Find any other BLAST data services that complement NCBI BLAST's coverage?

We could also imagine extending these single-source-biased service queries to more sophisticated ones that cover multiple services. For example, a user interested in sports medicine may want to discover services that are most similar to both PubMed and the sports-related site ESPN. Additionally, a user may simply be interested in discovering any non-obvious relationships that may exist among a group of data intensive web services.

Currently, there are no effective means to answer such service-centric queries without relying on significant human intervention or hand-tuned categorization schemes. Search engines are not designed to handle service-level queries. Similarly, a Yahoo!-style directory service offers general categories of data services but it does not provide service coverage and service specialty ratings. With the rapid increase in the number and variety of data intensive web services on both the Deep Web and the emerging web services paradigm, there is a growing demand for providing a service-centric framework for discovering and ranking data services.

To answer these challenges, we present a novel *source-biased* approach to automatically discover and rank relevant data intensive web services. It supports a service-centric view of the Web through source-biased probing and source-biased relevance detection and ranking metrics. Concretely, our approach is capable of answering source-centric queries of the form posed above by focusing on the nature and degree of the topical relevance of one service to others. Given a service like PubMed – called the *source* – our source-biased probing technique leverages the summary information of the source to generate a series of biased probes to other services – called the *targets*. This source-biased probing allows us to determine in very few interactions whether a target service is relevant to the source by probing the target with very focused probes. We introduce the *biased focus* metric to dis-

cover highly relevant data services and measure relevance between services. For example, we may discover that one service is more generic than another, more specialized than another, or complementary to another service. In addition to determine the nature and degree of the relevance of one service to another, we also employ the biased focus metric to rank target services. A target service with higher affinity to the source of bias will be ranked higher. We demonstrate how a simple implementation of our biased-probing technique outperforms alternative methods. To further reduce the number of similar probes, we introduce a performance enhancement to our basic approach, called *source-biased probing with focal terms*. The main idea is to identify terms in the unbiased summary of the source that share a similar topical category and then to divide the source summary into k clusters, where each cluster represents a group of similar summary terms. Our experiments show that source-biased probing with focal terms has the same success rate in discovering and ranking data intensive web services using fewer source-centric probes. Furthermore, we use source-biased probing and the biased focus measure as our fundamental building blocks to extend the source-biased service discovery framework to a more generalized service neighborhood graph model. We discuss the semantics of the neighborhood graph model, how we may reason about the relationships among multiple services, and how we may rank services based on this service neighborhood graph model. We also report initial experiments to show the effectiveness of our approach.

2. SYSTEM MODEL AND PROBLEM STATEMENT

In this paper, a *data intensive web service* is a web service that satisfies the following two requirements: (1) the service must provide access to its underlying information repository (typically through a request-response mechanism); and (2) the service is defined by the data it owns, not by the interface it provides.The first requirement is simply for practicality; a service without access is essentially off-line and invisible for our purposes. The second requirement merely emphasizes that what constitutes a data intensive web service is the data it owns. Two common examples are the Deep Web and web services.

Both the Deep Web and web services are underrepresented (or not represented at all) on popular search engines. A user interested in finding a particular Deep Web site may rely on a Yahoo!-style categorization scheme like the ones provided by InvisibleWeb and CompletePlanet [13, 5]. Web services are currently listed in unorganized UDDI-based directories. In both cases, a service-centric query of the form "What other data intensive web services are most similar to X?" is impossible to answer without significant human intervention. The large and increasing number of web services available on the Web today demands a service-centric view of the Web and an efficient and effective framework for automated service discovery and service ranking.

2.1 Modeling Services with Service Summaries

We consider a universe of discourse $\mathcal{W}$ consisting of D data intensive web services: $\mathcal{W} = \{S_1, S_2, \ldots, S_D\}$ where each service produces a set of documents in response to a particular service request. Hence, we describe each web service S_i as a set of M_i documents: $S_i = \{doc_1, doc_2, \cdots, doc_{M_i}\}$. There are N terms (t_1, t_2, ..., t_N) in the universe of discourse $\mathcal{W}$, where common stopwords (like 'a', 'the', and so on) have been eliminated. Optionally, the set of N terms may be further refined by stemming [20] to remove prefixes and suffixes.

Adopting a vector-space model [23, 24] of the service contents, we may describe each web service S_i as a vector consisting of the terms in the service along with a corresponding weight: [1]

$$\textsc{Summary}(S_i) = \{(t_1, w_{i1}), (t_2, w_{i2}), \cdots, (t_N, w_{iN})\}$$

A term that does not occur in any documents served by a service S_i will have weight 0. Typically, for any particular service S_i, only a fraction of the N terms will have non-zero weight. We refer to the number of non-zero weighted terms in S_i as N_i.

We call the vector $\textsc{Summary}(S_i)$ a *service summary* for the data intensive web service S_i. A service summary is a single aggregate vector that summarizes the overall distribution of terms in the set of documents produced by the service. To find $\textsc{Summary}(S_i)$, we must first represent each document doc_j ($1 \leq j \leq M$) as a vector of terms and the frequency of each term in the document:

$$doc_j = \{(t_1, freq_{j1}), (t_2, freq_{j2}), \cdots, (t_N, freq_{jN})\}$$

where $freq_{jk}$ is the frequency of occurrence of term t_k in document j. Initially the weight for each term may be based on the raw frequency of the term in the document, though there are alternative occurrence-based metrics like the normalized frequency of the term and the term-frequency inverse document-frequency ($TFIDF$) weight. TFIDF weights the terms in each document vector based on the characteristics of all documents in the set of documents.

Given a particular encoding for each document, we may generate the overall service summary in a number of ways. Initially, the weight for each term in the service summary may be based on the overall frequency of the term across all the documents in the service (called the *service frequency*, or *servFreq*): $w_{ik} = servFreq_{ik} = \sum_{j=1}^{M} freq_{jk}$. Alternatively, the weight for each term may be based on the number of documents in which each term occurs (called the *document count frequency*, or *docCount*): $w_{ik} = docCount_{ik} = \sum_{j=1}^{M} \mathcal{I}_j(t_k)$ where $\mathcal{I}_j(t_k)$ is an indicator function with value 1 if term t_k is in document j and 0 otherwise.

Once we have chosen our service model, to effectively compare two data intensive web services and determine the relevance of one service to another, we need two technical components: (1) a technique for generating a service summary; and (2) a metric for measuring the relevance between the two services.

2.2 Estimating Service Summaries

Ideally, we would have access to the complete set of documents belonging to a data intensive web service. We call a service summary for S_i built on these documents an *actual service summary* or $\textsc{ASummary}(S_i)$. However, most services do not (or will not) publish their entire set of documents for public consumption; rather, documents are accessible only through a request-response mechanism. Many

[1]This vector-based approach has been popularized in the database community by GlOSS [10] and related projects.

documents are created on-the-fly in response to a particular user query at a service. Hence, the documents themselves are transient objects that may not be consistent from moment-to-moment. Furthermore, the enormous size of the data repositories underlying many services coupled with the non-trivial costs of collecting documents (through repetitive requests and document download) makes it unreasonable to expect to completely capture every document available through a service. As a result, previous researchers have introduced several techniques for *probing* a service to generate a representative summary based on a small sample of the entire services [1, 2]. We call such a representative summary an *estimated service summary*, or ESUMMARY(S_i):

$$\text{ESUMMARY}(S_i) = \{(t_1, w_{i1}), (t_2, w_{i2}), \cdots, (t_N, w_{iN})\}$$

The number of occurring terms (i.e. those terms that have non-zero weight) in the estimated summary is denoted by N_i'. Typically, N_i' will be much less than the number of non-zero weighted terms N_i in the actual service summary since only a fraction of the total documents in a service will be examined. Hence, the goal of a prober is typically to find ESUMMARY(S_i) such that the relative distribution of terms closely matches the distribution of terms in ASUMMARY(S_i), even though there will be far fewer occurring terms in the estimated summary than in the actual summary.

We will discuss the details of these probing techniques in great detail shortly, but for now, let us suppose that we have a mechanism for approximating a service summary with ESUMMARY(S_i).

Several sampling techniques have been proposed to estimate data source summaries. All aim at estimating the overall summary of the data content served by a web service. We classify them into two categories: Random sampling and query-based sampling.

Random Sampling – No Bias

If we had unfettered access to a data intensive web service, we could randomly select terms from the service to generate the estimated service summary ESUMMARY(S_i). Barring that, we could randomly select documents with which to base the estimated service summary. We will call such a random selection mechanism an *unbiased prober* since all terms (or documents) are equally likely to be selected. In practice, an unbiased prober is unrealistic since most services only provide a query-based request-response mechanism for extracting documents. The request-response mechanism introduces bias through the ranking of returned documents and by providing incomplete access to the entire service. Hence, true unbiased selection is infeasible in practice.

Query-based Sampling – Query Bias

As a best approximation to unbiased probing, Callan et al. [1, 2] have introduced a query-based sampling technique for generating accurate estimates of data services. Their goal is to generate an estimated summary that matches the actual contents of the database by examining only a fraction of the total documents. The Callan technique relies on repeatedly requesting documents from a source using a limited set of queries. Since the documents extracted are not chosen randomly, but are biased by the querying mechanism, we say that the Callan technique displays *query bias*.

There are several ways to define the limited set of queries. A simplest one is to randomly select a limited number of keywords in a general dictionary. One can also start with a randomly selected keyword and then choose the subsequent keywords from the documents returned by the data service. More recently, Gravano et al. [16] have introduced an extension to the Callan-style probing technique that relies on a set of query probes learned by a classifier. Their probing method relies on a Yahoo!-style categorization hierarchy and requires learning a set of probes specific to the hierarchy. Probes are intended to guide placement of each service with the appropriate node of the classification hierarchy. This *classifier-based query probing* skews the service summaries towards classifier-determined topics. This method aims at classifying services and is effective at comparing two services based on a given categorization hierarchy, but it is not suitable for direct comparison of one service (source) to another (target) in terms of the content coverage of the target with respect to the source.

2.3 Comparing Service Summaries: Potential Problems

In order to determine the relevance of one data intensive web service S_i to another service S_j, we require an appropriate relevance metric. There are a number of possible relevance metrics to compare two service summaries. A fairly simple and straightforward approach is based on a count of the number of common terms in the two services S_i and S_j:

$$relevance(S_i, S_j) = \frac{|\text{ESUMMARY}(S_i) \cap \text{ESUMMARY}(S_j)|}{max(|\text{ESUMMARY}(S_j)|, |\text{ESUMMARY}(S_i)|)}$$

Two services with exactly the same terms represented in their estimated summaries will have $relevance(S_i, S_j) = 1$, indicating the highest possible degree of relevance. Conversely, two services with no terms in common will have $relevance(S_i, S_j) = 0$, indicating the lowest possible degree of relevance.

We now use an example to illustrate why the existing service summary estimation techniques are inadequate for effectively revealing interesting relationships between two data services, especially in terms of the content coverage of one (target) in the context of the other (source).

Example: *We collected fifty documents from the Google web service, the PubMed web service, and ESPN's search site, respectively, using an effective probing technique for service summary estimation [which we will discuss in detail shortly]. Based on real-world experience, we know that PubMed is exclusively an excellent service for health information. Google also is a great service for health information, although it does provide access to many other kinds of information. In contrast, ESPN is a sports-only site and of little relevance to PubMed. Using the service summaries constructed, we find that* $relevance(Google, PubMed) = 0.05$ *and* $relevance(ESPN, PubMed) = 0.06$. *In both cases the service summaries share very few terms in common and hence both Google and ESPN appear to be irrelevant with respect to PubMed. Based on these figures, we could incorrectly conclude two facts: (1) Google is irrelevant to PubMed; and (2) Relatively speaking, ESPN is more relevant to PubMed than Google.*

This example underlines two critical problems with current techniques for probing and comparing service summaries:

First, current service summary estimation techniques are concerned with generating *overall* summaries of the underlying data services. The goal is to generate essentially an unbiased estimate of the actual service summary. In this example, since Google has such broad coverage, very few

terms in an unbiased estimated summary may be common to the PubMed estimated service summary. Hence, many topics that are relevant for context based service comparisons may be under-represented or overlooked completely, since the summaries contain just a small fraction of the total terms in each service.

Second, the current relevance comparison metrics are concerned with how much content one service covers over the combined content served by the two services. This type of metric is good at capturing the size difference of two services in terms of data content they serve but fails to indicate the interesting relationship between two services in terms of the content coverage of one service (target) with respect to the other (source). Therefore, we need a relevance metric that can describe the nature and degree of the relationship between two services. In the context of our example, it would be interesting to discover both that Google is much more relevant to PubMed than ESPN and that Google has much broader coverage than PubMed. When comparing two services using estimated summaries, a relevance metric like the one described above is clearly inadequate.

Bearing these issues in mind, we propose a source-biased approach to service discovery and ranking that uses a source-biased query probing to generate biased estimated summaries and a biased-focus measure for better capturing the relationships between two services. The source-biased probing biases the service summary of a target service towards the service summary of the source, effectively revealing the nature and degree of the relationship between the source service and the target service. The biased-focus metric measures the similarity of the data content served by the target service in terms of the estimated summary of the source, successfully highlighting the importance of source-biased similarity in comparing services. Our experiments show that our source-biased approach is effective in revealing interesting relationships among services and efficient for discovering and ranking services in a source-specific context (see Section 6 for details).

3. SOURCE-BIASED SERVICE DISCOVERY: THE BASIC ALGORITHM

With the continued increase in the number of web services and the rapid growth in the amount of dynamic content served from the Deep Web, the problem of discovering and ranking services is becoming increasingly important. One way to discover interesting web services is to start from a given source service and try to find those target services that are highly relevant to the source. We refer to this type of service discovery as source-biased service discovery.

There are two fundamental steps in performing source-biased service discovery. First, in order to find the target data services that have high relevance to a source, we need to generate a source-biased summary of the target services instead of using unbiased summaries of the targets. We propose a source-biased probing algorithm that can compute the relevance of the target services with respect to the source in very few probes. Second, we need an efficient mechanism to measure the source-biased relevance of target services with respect to the source. We propose a so-called *biased focus* metric, taking a source-centric view of the target services. We next discuss each of these two steps in detail.

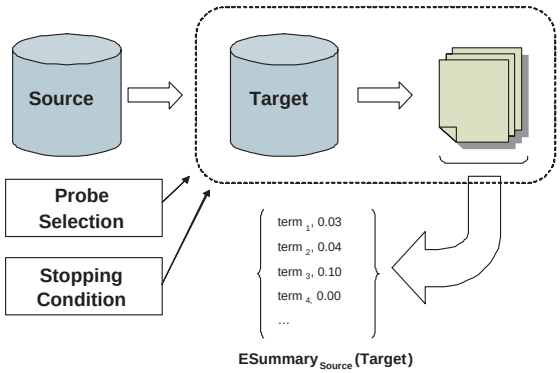

Figure 1: Source-Biased Probing

3.1 Source-Biased Probing

Given a service – called the *source* – the source-biased probing technique leverages the information available within the source to generate a series of biased probes to send to another service – called the *target*. This source-biased probing allows us to determine in very few interactions whether a *target* service is relevant to the *source* by probing the target with highly focused probes.

To help differentiate the source-biased approach from others discussed in Section 2, in this section we use σ to denote the source service and τ to denote the target service instead of S_i and S_j. The source-biased probing algorithm works as follows: We use the unbiased service summary of the source, denoted by $\text{ESUMMARY}(\sigma)$, as a dictionary of candidate probe terms. We then send a series of probe terms, selected from $\text{ESUMMARY}(\sigma)$, to the target service τ until a stopping condition is met. The final product of the source-biased probing is a service summary for τ that is biased towards σ. We denote this source-biased summary of the target service by $\text{ESUMMARY}_\sigma(\tau)$, where:

$$\text{ESUMMARY}_\sigma(\tau) = \{(t_1, w_1^\sigma), (t_2, w_2^\sigma), \cdots, (t_{N'}, w_N^\sigma)\}$$

For any target service τ, let $\text{ESUMMARY}(\tau)$ denote its unbiased summary obtained using either random sampling or query sampling as discussed earlier:

$$\text{ESUMMARY}(\tau) = \{(t_1, w_1), (t_2, w_2), \cdots, (t_{N'}, w_N)\}$$

It is important to note that typically the inequality $w_j \neq w_j^\sigma$ does hold. To distinguish the term weight w_j from the corresponding term weight in the biased target summary, we denote the bias by w_j^σ. Figure 1 illustrates the source-biased probing process. A sketch of the source-biased probing algorithm is given in Figure 2. The performance and effectiveness of the algorithm depends upon a number of factors, including the selection criterion used for choosing source-specific candidate probe terms, and the type of stop condition used to terminate the probing process.

In Figure 1, The arrow indicates that probes are drawn from the source σ and sent to the target τ for generating source-biased summary estimate $\text{ESUMMARY}_\sigma(\tau)$. In the rest of the paper, we sometimes use the shorthand $\sigma \rightarrow \tau$ to indicate that σ is used as a source of bias to probe a target service τ.

Now let us use the following example to illustrate the power of source-biased probing. For simplicity we are considering a simplistic world of only very few terms per service

SourceBiasedProbing(Service σ, Service τ)
For target service τ, initialize $\text{ESUMMARY}_\sigma(\tau) = \emptyset$.
repeat
Invoke the probe term selection algorithm to select a one-term query probe q from the source of bias $\text{ESUMMARY}(\sigma)$.
Construct a service request for τ with query q.
Send the request to the target service τ.
Retrieve the top-m documents from τ.
Update $\text{ESUMMARY}_\sigma(\tau)$ with the terms and frequencies from the top-m documents.
until Stop probing condition is met.
return $\text{ESUMMARY}_\sigma(\tau)$

Figure 2: Source-Biased Probing Algorithm

summary. In reality, each service summary would consist of orders of magnitude more terms:

Example: *Suppose that our goal is to determine the relevance of Google to the health service PubMed. We have an estimated summary for PubMed:* $\text{ESUMMARY}(PubMed) = \{arthritis, bacteria, cancer\}$ *(where for simplicity we have dropped the term weights from the summary). Now suppose that Google provides access to only three types of information: health, animals, and cars. In the actual service summary for Google, each topic is represented by three terms:* $\text{ASUMMARY}(Google) = \{arthritis, bacteria, cancer, dog, elephant, frog, garage, helmet, indycar\}$. *If we were to estimate Google by using an unbiased prober, we would essentially randomly select terms from Google to serve as an approximation for the actual summary. Constraining the estimate to three terms, one such unbiased estimate could be:* $\text{ESUMMARY}(Google) = \{arthritis, frog, helmet\}$ *which shares only one term with PubMed. Hence we could conclude that Google is not relevant to PubMed. In contrast, if we were to use a PubMed-biased prober, we could focus on only those terms in Google that are relevant to PubMed by sending only health-related probes:* $\text{ESUMMARY}_{PubMed}(Google) = \{arthritis, bacteria, cancer\}$. *Hence, the source-biased probing accentuates the commonality between the two services.*

From Algorithm 2, it is clear that there are two components that have critical impact on the performance of the source-biased probing algorithm: the selection of query probing terms and the type of stop probing condition to use.

Mechanisms to Select Probe Terms
There are several possible ways to select the probes based on the statistics stored with each service summary, including uniform random selection and selection based on top-weighted terms. In general, the selection criterion will recommend a query term drawn from the set of all non-zero weighted terms in the unbiased source summary $\text{ESUMMARY}(\sigma)$.

Uniform Random Selection: In this simplest of selection techniques, each term that occurs in $\text{ESUMMARY}(\sigma)$ has an equal probability of being selected, i.e.
$Prob(\text{selecting term } j) = \frac{1}{N}$.

Weight-Based Selection: Rather than randomly selecting query terms, we could instead rely on a ranking of the terms by one of the statistics that are stored with each service summary. For example, all terms in $\text{ESUMMARY}(\sigma)$ could be ranked according to the weight of each term. Terms would then be selected in descending order of weight. Depending on the type of weight stored (e.g. $servFreq$, $docCount$, etc.), several flavors of weight-based selection may be considered.

Different Types of Stop Probing Conditions
The stop probing condition is the second critical component in the source-biased probing algorithm. We consider four different types of conditions that might be used in practice:

Number of Queries: After some fixed number of query probes ($MaxProbes$), end the probing. This condition is agnostic to the number of documents that are examined for each service.

Documents Returned: In contrast to the first technique, the second condition considers not the number of queries, but the total number of documents ($MaxDocs$) returned by the service. Since some queries may return no documents, this stopping condition will require more query probes than the first alternative when $MaxProbes = MaxDocs$.

Document Thresholding: Rather than treating each document the same, this third alternative applies a threshold value to each document to determine if it should be counted toward $MaxDocs$. For each document, we may calculate the relevance of the document to the source of bias $\text{ESUMMARY}(\sigma)$. If the document relevance is greater than some threshold value, then the document is counted. Otherwise, the document is discarded.

Steady-State: Rather than relying on a count of queries or documents, this final stopping condition alternative instead relies on the estimated summary reaching a steady-state. After each probe, we calculate the difference between the new value of $\text{ESUMMARY}_\sigma(\tau)$ and the old value. If the difference (which may be calculated in a number of ways) is less than some small value ϵ, then we consider the summary stable and stop the probing.

In the experiments section, we report the results of varying both of these key parameters.

3.2 Evaluating Relevance: Biased Focus

We previously asserted that to effectively determine the relevance of one service to another, we would require both a technique for generating a source-biased service summary and a source-biased metric for measuring the relevance of a target service in the context of the source. In the previous section, we presented the algorithm and the factors that impact the performance of the algorithm for generating source-biased summary estimates of a data service. In this section we present several flavors of a *biased focus* measure for better understanding the relationship between a source and a target.

Given a source service and a target service, we begin by discussing the requirements of an appropriate biased focus measure and then discuss several possible variations.

DEFINITION 1 (BIASED FOCUS). *Let σ denote a source service modeled by an unbiased summary and τ denote a target service with a σ-biased summary, and let $focus_\sigma(\tau)$ denote the source-biased focus measure. We define $focus_\sigma(\tau)$ to be a measure of the topical focus of the target service τ with respect to the source of bias σ. The focus metric ranges from 0 to 1, with lower values indicating less focus and higher values indicating more focus.*

In general, $focus$ is not a symmetric relation. Hence

we may describe any two services σ and τ with the focus in terms of σ (i.e. $focus_\sigma(\tau)$) and in terms of τ (i.e. $focus_\tau(\sigma)$).

There are several ways to calculate the biased focus for a source and a target. One approach is to approximate the focus measure by computing the ratio of common terms between source and target over the source summary estimate. We call this method the common-term based focus measure, denoted by $CTfocus_\sigma(\tau)$.

$$CTfocus_\sigma(\tau) = \frac{|\text{ESUMMARY}(\sigma) \cap \text{ESUMMARY}_\sigma(\tau)|}{|\text{ESUMMARY}(\sigma)|}$$

This approximation counts the number of common terms between the source of bias and the target and divides by the size of the source of bias. So if all terms in the source of bias occur in the target, then the target is perfectly focused on the source and $CTfocus_\sigma(\tau) = 1$. Conversely, if no terms in the source of bias occur in the target, then the target has no focus on the source and $CTfocus_\sigma(\tau) = 0$. Unfortunately, a common-term based focus measure will tend to understate the importance of highly-weighted terms and overvalue the importance of lowly-weighted terms. Consider the following example:

Example: Let σ be a source service and its associated service summary be given below: $\text{ESUMMARY}(\sigma) = \{(a, 100), (b, 1), (c, 1), (d, 1), (e, 1)\}$. *Assume that we also have two source-biased summaries for targets τ_1 and τ_2 that are generated by source σ-biased probing queries:* $\text{ESUMMARY}_\sigma(\tau_1) = \{(a, 100)\}$ *and* $\text{ESUMMARY}_\sigma(\tau_2) = \{(b, 1), (c, 1), (d, 1), (e, 1), (f, 100)\}$ *Calculating the common term-based focus, we find $CTfocus_\sigma(\tau_1) = 0.2$ and $CTfocus_\sigma(\tau_2) = 0.8$. So even though τ_2 is heavily weighted towards the term f, it is considered more relevant to σ than τ_1.*

An obvious solution to address the above-mentioned problem is to use the term-weight based focus measure, denoted by $TWfocus_\sigma(\tau)$:

$$TWfocus_\sigma(\tau) = \frac{\sum_{k \in \text{ESUMMARY}_\sigma(\tau)} w_{\sigma k}}{\sum_{k \in \text{ESUMMARY}(\sigma)} w_{\sigma k}}$$

where $w_{\sigma k}$ is the weight for term k in ESUMMARY_σ. The term weight based focus measure can be seen as a generalization of the ctf_{ratio} introduced in [1].[2]

While the $TWfocus_\sigma(\tau)$ approximation overcomes the problems of the $CTfocus_\sigma(\tau)$, it introduces new issues. For example, the term weights used in the $TWfocus_\sigma(\tau)$ approximation are from the unbiased summary of the source. Thus it does not give a good estimate on the source-biased measure.

We propose to use the well-known cosine similarity (or normalized inner product) to approximate the source-biased focus measure. We define the cosine-based focus as follows:

$$Cosine\_focus_\sigma(\tau) = \left(\frac{\sum_{k=1}^{N} w_{\sigma k} w_{\tau k}^{\sigma}}{\sqrt{\sum_{k=1}^{N} (w_{\sigma k})^2} \cdot \sqrt{\sum_{k=1}^{N} (w_{\tau k}^{\sigma})^2}} \right)$$

where $w_{\sigma k}$ is the weight for term k in $\text{ESUMMARY}(\sigma)$ and $w_{\tau k}^{\sigma}$ is the σ-biased weight for term k in $\text{ESUMMARY}_\sigma(\tau)$. The cosine ranges from 0 to 1, with higher scores indicating a higher degree of similarity. In contrast, the cosine between orthogonal vectors is 0, indicating that they are completely dissimilar. The cosine measures the angle between two vectors, regardless of the length of each vector. Intuitively, the cosine-based biased focus is appealing since it more reasonably captures the relevance between two services.

The biased focus metrics is not only used for measuring service relevance but also used for ranking the set of target services $\tau_1, \tau_2, \ldots, \tau_N$ with respect to a source σ. In Section 5, we shall show how to use the biased-focus for drawing interesting inferences about the relationships between different but relevant services.

4. SOURCE-BIASED PROBING WITH FOCAL TERMS

Recall the discussions in the previous section, one of the critical parameters to the success of source-biased probing is the choice of probe terms from the source of bias σ. We discussed several random selection techniques as well as different ways to define stop-probing conditions. In all cases considered so far, the estimated source summary is treated as a single repository of candidate probe terms. In this section we introduce a refinement over these simple selection techniques whereby the source summary is segmented into k groups of co-occurring terms. The main idea is to iteratively select one term from each of the k groups to probe the target. We call this term the focal term of the corresponding group. When used in conjunction with the general source-biased probing algorithm, we have an enhanced version called *source-biased probing with focal terms*. Like the basic algorithm of source-biased probing, the goal remains to produce source-biased target service summaries that are effective for detecting interesting relationships between a source of bias and a target service. A unique advantage of using focal terms is that these source-biased summaries of target services can be generated in far fewer queries and with higher quality. We below describe this enhanced source-biased probing algorithm in detail and will show the benefit of this algorithm through experiments in Section 6.

4.1 Focal Terms and Focal Term Groups

Let σ denote a source service with its unbiased service summary ESUMMARY_σ. We denote the set of terms with non-zero weight in ESUMMARY_σ (i.e. the terms that actually occur in the service σ) as $Terms(\sigma)$, where $Terms(\sigma)$ consists of n terms t_1, t_2, ..., t_n.

A *focal term group* is a subset of terms in the set $Terms(\sigma)$ that co-occur in the documents of σ. We denote a focal term group i as $FTerms_i$. The main idea behind source-biased probing with focal terms is to partition the set $Terms(\sigma)$ into k disjoint term groups such that the terms within each term group co-occur in documents of σ more frequently than they do with terms from other term groups.

Formally, we need an algorithm that can find a partition of $Terms(\sigma)$ into k focal term groups:

$$T_k^{erms}(\sigma) = \{FTerms_1, \ldots, FTerms_i, \ldots, FTerms_k| \bigcup_{i=1} FTerms_i = \{t_1, ..., t_n\} \text{ and } FTerms_i \cap FTerms_j = \emptyset\}$$

In Table 1, we show an example of five focal term groups for a collection of 100 PubMed documents. Note that k

[2]The ctf_{ratio} is presented in the context of comparing an estimated database summary DB' to an actual database summary DB. $ctf_{ratio} = \sum_{i \in DB'} ctf_i / \sum_{i \in DB} ctf_i$, where ctf_i = number of times term i occurs in the source. Here, we have generalized this formulation for comparison of summaries from different services, and for use with term weightings other than the ctf.

Table 1: Example Focal Terms for PubMed

Term Group	Terms
1	care, education, family, management, ...
2	brain, gene, protein, nucleotide, ...
3	clinical, noteworthy, taxonomy, ...
4	experimental, molecular, therapy, ...
5	aids, evidence, research, winter, ...

is intended to be very small since the focal term groups are meant to be very coarse. We will describe the concrete algorithm to find k partitions of the set $Terms(\sigma)$ in the next section.

Given k focal term groups, by selecting a focal term from each term group $FTerms_i$ as a probing query, we hope to retrieve documents that also contain many of the other words in that focal term group. For example, suppose we are using a frequency-based measure for query probe selection from PubMed. The top four query terms may be "brain", "gene", "protein", and "nucleotide". Suppose these four terms tend to co-occur with each other as indicated in Table 1. By sending the first query "brain" to a target service, we could reasonably expect to find the other three terms since our analysis of the source indicates that these four terms tend to co-occur. A naive source-biased prober would ignore this co-occurrence information and, instead, send the other three queries "gene", "protein", and "nucleotide", even though we might reasonably expect for those queries to generate documents similar to the first query "brain". In essence, we will have used four queries when a single query would have sufficed at adequately exploring the term space of the target.

The sophistication of source-biased probing with focal terms is to identify these co-occurrence relationships in order to reduce the number of queries necessary to efficiently detect relationships between a source and a target service. By using focal terms, we may generate more accurate biased summaries of target services in far fewer probe queries and with higher quality.

In an ideal case, every focal term group would consist of terms that only co-occur with each other and not with any other terms in the other focal terms groups. By selecting a single term from each perfectly segmented term group, we ideally could send no more than k probes, one for each focal term group. Each probe would produce a document that contained every other term in that focal term group. In the more realistic setting, we will need to handle varying degrees of co-occurrence, but we still expect a good reduction in the number of probes necessary to generate a high-quality biased summary estimate for each target service.

It is important to note that, unlike previous research in grouping terms – for query-expansion [28, 22] or finding similar terms [25] – our goal is not to find close semantic relationships between terms, but rather to find very coarse co-occurrence associations among terms to support a more efficient and effective biased service summary estimation. For example, though we may discover that "brain" and "protein" tend to co-occur in a service, we do not claim that there is a close semantic relationship between the two terms.

4.2 Finding Focal Terms

Now that we have discussed the motivation of finding focal terms, we are still faced with the task of actually segmenting $Terms(\sigma)$ into k groups of focal terms. In this section, we discuss how we may adapt a popular clustering technique to the problem of focal term discovery.

Recall Section 2.1, we view a data intensive web service S_i as a set of documents, each of which is described by a vector of terms and weights. We now invert our view of a service using the same set of information. We consider a service S_i as a collection of *terms*, each of which is described by a vector of the documents in which the term occurs and a weight describing the occurrence frequency of the term in the corresponding document. Hence, we have:

$$Terms(S_i) = \{term_1, term_2, \cdots, term_N\}$$

For the N terms in the service, each $term_j$ $(1 \leq j \leq N)$ is a vector of documents and weights:

$$term_j = \{(doc_1, w_{j1}), (doc_2, w_{j2}), \cdots, (doc_M, w_{jM})\}$$

We can define a segmentation technique for finding focal term groups by clustering the set $Terms(S_i)$ into k clusters. Given the term vectors of services, and the similarity function, a number of clustering algorithms can be applied to partition the set $Terms(S_i)$ of N terms into k clusters. We choose Simple K-Means since it is conceptually simple and computationally efficient. The algorithm starts by generating k random cluster centers. Each term is assigned to the cluster with the most similar (or least distant) center. The similarity is computed based on the closeness of the term and each of the cluster centers. Then the algorithm refines the k cluster centers based on the centroid of each cluster. Terms are then re-assigned to the cluster with the most similar center. The cycle of calculating centroids and assigning terms in $Terms(S_i)$ to k clusters repeats until the cluster centroids stabilize. Let C denote both a cluster and the set of terms in the cluster. The centroid of cluster C is:

$$centroid_C = \left\{ \begin{array}{c} (doc_1, \frac{1}{|C|} \sum\limits_{i \in C} w_{j1}) \\ (doc_2, \frac{1}{|C|} \sum\limits_{i \in C} w_{j2}) \\ \cdots \\ (doc_M, \frac{1}{|C|} \sum\limits_{i \in C} w_{jM}) \end{array} \right\}$$

where w_{jl} is the weight of term j in document l, and the formula $\frac{1}{|C|} \sum_{l \in C} w_{jl}$ denotes the average weight of the document l in the cluster C. A sketch of the K-Means term clustering based on term-vector of a service is provided in Figure 3.

The similarity function used in Figure 3 can be defined using a number of functions. In this paper, we use the cosine similarity function. Given a set of N terms and a set of M documents, where w_{ik} denotes the weight for term k in document i $(1 \leq k \leq N,\ 1 \leq i \leq M)$, the cosine function prescribes:

$$sim(term_i, term_j) = \left(\frac{\sum_{k=1}^{N} w_{ik} w_{jk}}{\sqrt{w_{ik}^2} \cdot \sqrt{w_{jk}^2}} \right)$$

4.3 Selecting k Source-biased Probes

Once the k focal term groups have been constructed for a source, the remaining problem is how to select the best k terms for probing a target service. We propose a simple

FocalTerms(Number of Clusters k, Input Vectors $\mathcal{D}$)
Let $\mathcal{D} = \{d_1, ..., d_n\}$ denote the set of n term vectors
Let M denote the total number of documents in $\mathcal{D}$
Let $d_j = < (doc_1, w_j1), \ldots, (doc_M, w_{jM}) >$ denote a term vector of M elements, w_{jl} is the TFIDF weight of the doc_l in term j $(l = 1, \ldots, M)$
Let $\mathcal{C} = \{C_1, ..., C_k\}$ denote a clustering of $\mathcal{D}$ into k clusters.
Let μ_i denote the center of cluster C_i
foreach cluster C_i
  Randomly pick a term vector, say d_j from $\mathcal{D}$
  Initialize a cluster center $\mu_i = d_j$, where $d_j \in \mathcal{D}$
repeat
  foreach input term vector $d_j \in \mathcal{D}$
    foreach cluster $C_i \in \mathcal{C}$ $i = 1, \ldots, k$
      compute $\delta_i = sim(d_j, mu_i)$
    if δ_h is the smallest among $\delta_1, \delta_2, \ldots, \delta_k$
      mu_h is the nearest cluster center to d_j
    Assign d_j to the cluster C_h
  // *refine cluster centers using centroids*
  foreach cluster $C_i \in \mathcal{C}$
    foreach doc l in d_j $(l = 1, \ldots, M))$
      $cw_{ij} \leftarrow \frac{1}{|C_i|} \sum_{l=1}^{M} w_{jl}$
    $\mu_i \leftarrow < (doc_1, cw_{i1}), \ldots, (doc_M, cw_{iM}) >$
until cluster centers no longer change
return $\mathcal{C}$

Figure 3: Focal Term Clustering Algorithm

round-robin selection technique whereby a single term is selected from each focal term group in turn. Once a single term has been selected from each group, the cycle repeats by selecting a second term from each group, a third term, and so on.

Given this basic strategy, we may use a number of techniques for determining the order by which to select terms from the k groups and for selecting probe terms from each focal term group. One way to determine the order of focal term groups is based upon the size of each group. We begin with the group with the most terms and end each cycle with the group that has the smallest number of terms. For each focal term group, we may decide which term to select for each cycle by using one of the selection criteria discussed in Section 3.

Now that we have discussed source-biased probing with focal terms, we next discuss how to build a service neighborhood using source-biased probing as a primitive and discuss how to infer interesting relationships among service nodes from the structure of the service neighborhood graph.

5. SERVICE NEIGHBORHOOD

In this section, we extend the basic source-biased probing algorithm to a service neighborhood graph of interconnected services. We show how services may be ranked relative to a source, how a neighborhood graph of interconnected data services may be constructed, and how other interesting relationships may be inferred from the neighborhood graph.

5.1 The Graph Model

We model the world of data intensive web services as a graph, where the nodes in the graph are services and the edges in the graph indicate biased probing from a source to a target:

DEFINITION 2 (SERVICE NEIGHBORHOOD GRAPH). *A service neighborhood graph is a weighted, directed graph $G = (V, E)$ where V is a finite set of data intensive web services and E is the edge set of G. Each edge $e \in E$ is an ordered set of vertices (u, v), where $u, v \in V$. Each edge has an associated weight, given by the weight function $w : E \rightarrow \mathscr{R}$. The weight $w(u, v)$ of the edge $(u, v) \in E$ is given by $focus_u(v)$.*

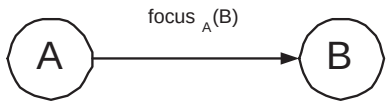

Figure 4: Simple Service Graph

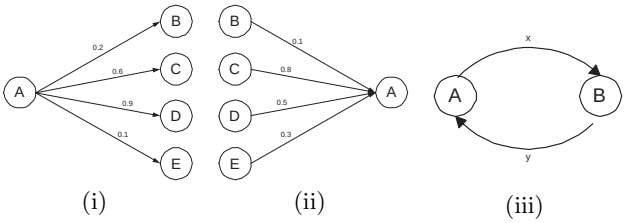

Figure 5: Reasoning about Service Neighborhoods with Source-Biased Probing

As an example, we show in Figure 4 a simple service neighborhood consisting of two services – A and B. The arrow from A to B indicates that the service A has been used as a source of bias to probe the target service B. The arrow is annotated with a weight for $focus_A(B)$.

For a universe of discourse $\mathcal{W}$ consisting of D data services: $\mathcal{W} = \{W_1, W_2, \ldots, W_D\}$, we may have a service neighborhood graph consisting of D nodes and up to $D(D-1)$ edges in the case that every service has served as a source of bias for every other service in the graph. In practice, we expect the number of edges to be much less, as many services may never be used as a source of bias.

5.2 Graph Semantics

Given the basic graph model, we now discuss several important inferences that may be made over the graph.

Ranking Target Services

In Figure 5(i), we show a single source of bias A and multiple target services B, C, D, and E. For each target service, we have an appropriate focus measure ($focus_A(B)$, $focus_A(C)$, $focus_A(D)$, and $focus_A(E)$) annotating the link from A to the target. We may then rank the target services in descending order in terms of their focus with respect to the source of bias A. In this case, we would order the target services D, C, B, E. In other words, the biased estimate of D contains 90% of the terms in A, whereas E contains only 10% of the terms in A.

Evaluating a Source of Bias

In Figure 5(ii), we reverse the situation and show a single target service A and multiple sources of bias B, C, D, and E. The single target A has one incoming arrow for each source of bias, annotated with the appropriate focus measure ($focus_B(A)$, $focus_C(A)$, $focus_D(A)$, and $focus_E(A)$). We may then rank the sources of bias in descending order in terms of their focus with respect to the target A. In this case, we would order the sources of bias C, D, E, B. In other words, the C-biased estimate of A contains 80% of

the terms in C, whereas the B-biased estimate of A contains only 10% of the terms in B.

Inferring Pairwise Relationships

In Figure 5(iii), we consider two services where each has served as a source of bias to probe the other. Let $focus_A(B) = x$. Let $focus_B(A) = y$. We define threshold values λ_{high} and λ_{low}, where $0 < \lambda_{low} < \lambda_{high} < 1$. These threshold values are intended to evaluate the quality of the relevance of one service to another as we illustrate next.

If $x > \lambda_{high}$ and $y > \lambda_{high}$, then we may conclude that A is sufficiently focused on B and B is sufficiently focused on A. Hence, the two services are approximately the same in terms of their content coverage. We call this approximate equality λ-equivalence to indicate that the equivalence is not total, but a function of the parameter λ_{high}.

If $x > \lambda_{high}$ and $y < \lambda_{low}$, then a sufficient portion of A is contained in B, but very little of B is contained in A. Hence, we may conclude that A is a λ-subset of B and B is a λ-superset of A, where again we use the λ prefix to indicate that A is not a strict subset, but rather that A's relationship to B is parameterized by λ.

If $x < \lambda_{low}$ and $y < \lambda_{low}$, then we may conclude that A and B are sufficiently concerned with different topics since neither is very focused on the other. We call this approximate inequality λ-disjointness to indicate that the disjointness is not total, but a function of the parameter λ_{low}.

6. EXPERIMENTS

In this section, we present experimental results to show how source-biased probing compares to competing techniques and how various parameters affect the behavior and performance of source-biased probing. Additionally, we illustrate the advantages of the two extensions to source-biased probing discussed above – namely, focal term based probing and the service neighborhood graph model.

6.1 Experimental Data and Setup

We evaluate source-biased probing over a dataset that is designed to emulate the diversity and scope of real-world data intensive web services and that also remains under our control for experimental validation.

We collected articles from 1,000 usenet newsgroups over the period June to July 2003. We randomly chose newsgroups from the top-level categories of bionet, comp, gnu, humanities, k12, misc, news, rec, sci, soc, and talk. We eliminated overly small newsgroups containing fewer than 100 articles, heavily spammed newsgroups, and newsgroups with primarily binary data. After filtering out these groups, we were left 593 newsgroups, representing over 2GB of data. Each newsgroup ranged in size from 100 to 16,000 articles, with an average of 1,400 articles per newsgroup. Since each of these newsgroups is focused on a single topic, we refer to these 593 newsgroups as the *single topic collection*.

In an effort to match the heterogeneity inherent in many real-world data intensive web services, we constructed 100 new groups by randomly combining articles from the single topic collection. For each new group, we randomly selected four collections from the single topic dataset to populate a new group with on average 1,400 articles. We refer to these 100 newsgroups as the *mixed topic collection*.

Finally, we created a third type of collection called the *aggregate collection*. Again, we aimed for a group average of 1,400 articles. For any group of three or more newsgroups with a common prefix from the single topic collection (e.g. `comp.databases.informix`, `comp.databases.oracle`, and `comp.databases.sybase`), we constructed a new aggregate collection (e.g. `comp.databases.aggregate`) containing randomly selected articles from each component newsgroup. This step resulted in 56 aggregate groups. To emulate a Google-like service with wide coverage, we created a single collection called 'root' that contained 10 random articles from each newsgroup in the single topic collection.

In total, the combined newsgroup collections consists of 2.5GB worth of articles in 750 groups.

We built a probing engine in Java 1.4 for use in all of our experiments. For each group in the dataset, we constructed the actual service summary based on the overall service frequency of each term ($servFreq$). We eliminated a set of common stopwords (e.g. 'a', 'the', and so on) as well as a specialized list of newsgroup-specific stopwords (e.g. 'wrote', 'said', 'reply', and so on). Terms were not stemmed.

6.2 Experimental Results

6.2.1 Probing Effectiveness and Efficiency

For our first set of experiments, we compare source-biased probing with three alternative probing techniques – two flavors of query-biased probers and an unbiased prober.

For the source-biased prober (*Source Bias*), we consider a basic version that uses a weight-based selection mechanism to choose the most highly weighted terms as probes. For this experiment we consider weights based on the overall service frequency ($servFreq$) of terms in the source's service summary. We evaluate alternative selection techniques in Section 6.2.2.

For the query-biased probers, we consider two versions. For the first (*Query Bias 1*), probe terms are randomly selected from the standard Unix dictionary of English terms. For the second (*Query Bias 2*), the initial probes are selected from the standard Unix dictionary, but once the first document has been retrieved from the target, all subsequent probes are selected based on the estimated $servFreq$ of the target service.

For the unbiased prober (*No Bias*), we consider a prober that selects documents at random from each target.

We selected 10 source services and 10 target services at random from the entire dataset, resulting in 100 source-target pairs. For each pair, we ran each of the four probing techniques, collecting a maximum of 5 documents per query from each target. We collected data for up to 100 total documents retrieved from each target. In Figure 6, we show the average $Cosine\_focus_\sigma(\tau)$ over all 100 source-target pairs as a function of the number of documents examined in each target. In this case, we can think of $Cosine\_focus_\sigma(\tau)$ as a proxy for the quality of the documents being extracted from each target. Since we are comparing the same set of targets across all four probing techniques, a higher $focus$ indicates that a prober has identified a more relevant set of documents than another. The *Source Bias* prober displays the highest $Cosine\_focus_\sigma(\tau)$ for all values of examined documents. The *No Bias* and *Query Bias 1* probers are bunched together below *Source Bias*. The lowest in all cases is *Query Bias 2*.

As you can see, the source-biased prober finds on average a higher degree of relevance between each target and source. The reason is that source-biased probing hones in on the

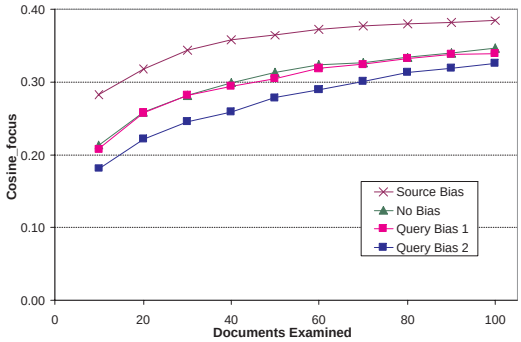

Figure 6: Comparing Four Probers

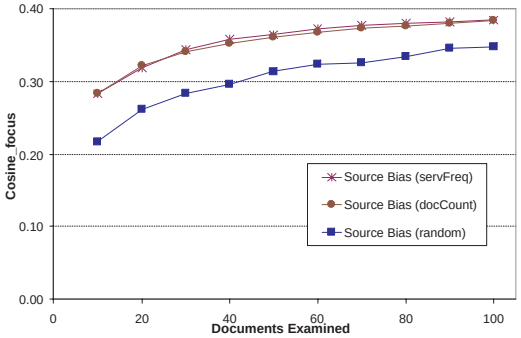

Figure 8: Query Selection Comparison

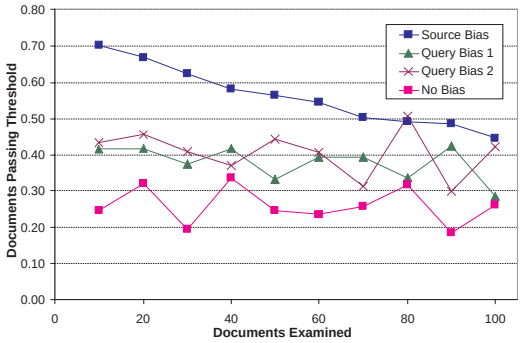

Figure 7: SBP Identifies Higher-Quality Documents

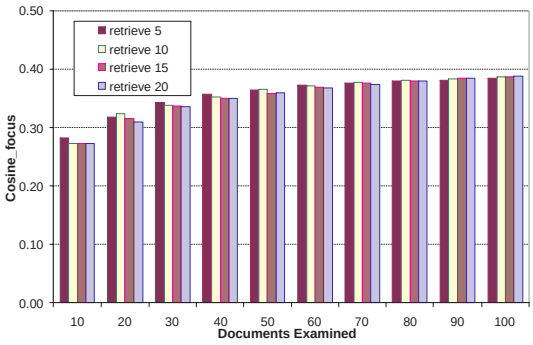

Figure 9: Documents Retrieved Comparison

most relevant documents between a given source and target.

To further explain why *Source Bias* outperforms the other probing techniques, consider Figure 7. Here we show the average quality of the documents examined for each of the four probers. For every ten documents examined, we calculate the fraction that pass a 0.1 $Cosine\_focus_\sigma(\tau)$ threshold with the source's unbiased service summary. The source-biased prober identifies documents of higher quality than the other techniques. As the number of documents examined increases, the source-biased approach begins to approach the other probers, since most of the quality documents have already been extracted.

Hence, we may conclude that source-biased probing is more effective at discerning relevant documents to a source of bias than are either a query-biased prober or an unbiased prober.

6.2.2 *Varying Key Parameters*

In this section, we discuss the impact of several parameters on the success of source-biased probing. Again, we selected 10 source services and 10 target services at random from the entire dataset, resulting in 100 source-target pairs. For each of these experiments, we consider only source-biased probing.

The first parameter we consider is the choice of query selection for source-biased probing. We consider three alternatives: random probe selection (*Source Bias (random)*), probe selection based on the overall service frequency of the source summary (*Source Bias (servFreq)*), and probe selection based on the document count of each term in the source summary (*Source Bias (docCount)*). We show the results in Figure 8.

Again, we can think of $Cosine\_focus_\sigma(\tau)$ as a proxy for the quality of the documents being extracted from each target. The two frequency-based measures result in approximately the same quality of extracted document. Interestingly, the random selection technique performs significantly worse. Since the set of candidate probes in a source's service summary is large, it seems reasonable to conclude that a random prober misses out on the critical discriminating terms.

The second parameter we consider is the number of documents retrieved for each query. We considered a basic version of source-biased probing using the overall service frequency as the probe selection mechanism. We vary the number of documents we retrieve, from 5 up to 20.

As you can see in Figure 9, there is little change, so varying retrieved documents appears not to have a significant impact on the quality of source-biased probing.

The third parameter we compare is the choice of focus measure. In Figure 10, we compare the three versions of focus first discussed in Section 3.2: $Cosine\_focus_\sigma(\tau)$, $TWfocus_\sigma(\tau)$, and $CTfocus_\sigma(\tau)$.

The dashed lines indicate the actual value of the overall focus measures calculated based on the actual service summaries of the sources and targets. The upper dashed line corresponds to the actual $TWfocus_\sigma(\tau)$. The other two dashed lines correspond to $Cosine\_focus_\sigma(\tau)$ and $CTfocus_\sigma(\tau)$ and are overlapping. The first critical point to note is that both the $TWfocus_\sigma(\tau)$ and $CTfocus_\sigma(\tau)$ are slow to ap-

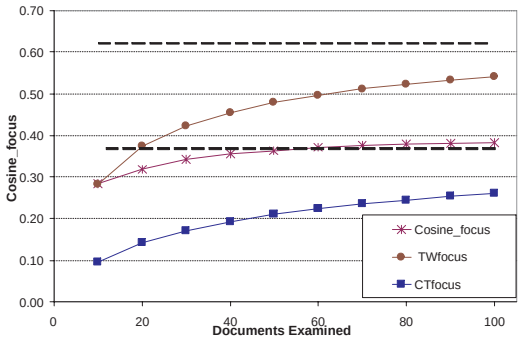

Figure 10: Comparison of Three Focus Measures

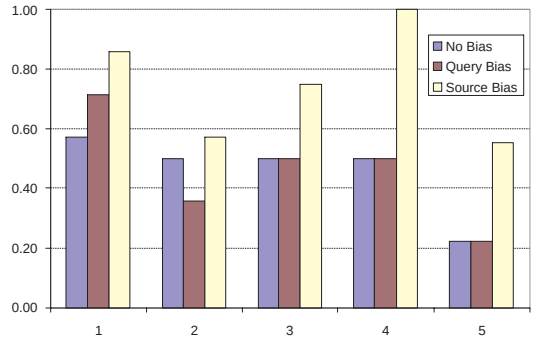

Figure 11: Comparison of Ranking Quality

proach the overall focus as indicated by the dashed lines. In contrast, the cosine-based focus approaches the actual focus in only 45 documents on average. Additionally, we note that $Cosine\_focus_{\sigma}(\tau)$ slightly overestimates the actual focus. This is reasonable, since for source-biased estimates based on very few documents, we would expect to identify high-quality documents first. Hence, the focus should be an overestimate. As the number of documents examined in a target nears the total available in the target, this upward bias should disappear.

6.2.3 Ranking Evaluation

As we have seen, source-biased probing is effective at identifying source-relevant documents in a target service. But how well does source-biased probing compare with the alternative techniques when it comes to ranking targets based on their relevance to a source? We randomly selected 5 sources to compare to the entire set of 750 groups. We compared the three probers *Source Bias*, *Query Bias 1*, and *No Bias* discussed above.

For each of the 5 sources, we identified the number of relevant groups in the entire dataset (call this total r for each source). For non-obvious cases, we determined relevance by the consensus opinion of three volunteers. We then evaluated the three probers by collecting 20 documents for each candidate target and ranking the targets by the $Cosine\_focus_{\sigma}(\tau)$ metric. We calculated the effectiveness for each source as the percentage of relevant targets ranked in the top-r.

As you can see, the source-biased approach outperforms both the query-biased and unbiased probers. Upon further inspection, we discovered that all probers performed approximately the same when comparing two nearly identical services (e.g. `comp.unix.admin` vs. `comp.unix.misc`) from the single topic collection. The source-biased approach far outstripped the others when evaluating a newsgroup from either the mixed topic collection or the aggregate collection. In these two cases, source-biased probing proved effective at honing in on relevant documents that an unbiased or query-biased prober might overlook.

6.2.4 Source-Biased Probing Extensions

In our final set of experiments, we illustrate the advantages of source-biased probing with focal terms and the service neighborhood graph model.

To evaluate source-biased probing with focal terms, we compared basic source-biased probing with random term selection versus two flavors of focal term probing – focal term probing with 3 focal term groups and with 5 focal term groups. For each of the focal term versions, we selected focal term groups in decreasing order of size. For each group, we randomly selected a probe term. We selected 5 source services and 10 target services at random from the entire dataset, resulting in 50 source-target pairs. For each query, we collected a maximum of 5 documents from each target for each query.

The two focal term techniques resulted in slightly higher quality documents being extracted for each target.

Finally, to illustrate the power inherent in the service neighborhood graph, we show in Table 2 a few sample inferences that can be made when services A and B are used to probe each other, and $Cosine\_focus_{\sigma}(\tau)$ is the focus metric. Consider the first pair of Mac-related newsgroups. For $\lambda_{high} = 0.70$, we would conclude that the two are λ-equivalent. In contrast, the sewing and perl groups are quite dissimilar, as indicated by the low focus values.

7. RELATED WORK

In the database community, considerable research has been dedicated to the *database selection* problem. [10, 11, 3, 6, 7, 8, 9, 12, 17, 18, 21, 29] In database selection, the problem is to take a query and match it to potentially relevant databases for processing. Typically the database exports a description to help guide database selection. Instead of matching a query to a set of services, we are concerned with grouping and ranking data services based on their relevance to a source of bias.

As mentioned before, our source-biased probing technique builds on the work of Callan et al. Other researchers have previously studied the problem of repeatedly querying an unknown database in an effort to generate a summary of the database internals [16, 14, 15, 1, 27, 2, 12, 26, 19, 4]. The main purpose of these techniques is to generate a representative content summary of the underlying database. Querying methods suggested include the use of random queries, queries learned from a classifier, and queries based on a feedback cycle between the query and the response.

More recently, Gravano et al. [16, 14] have introduced an extension to the Callan-style probing technique that relies on a learned set of queries for database classification. Their probing method is effective for classifying Deep Web sites into a pre-determined Yahoo!-style rigid hierarchy, but requires the potentially burdensome and inflexible task of

Table 2: Inferring Semantics

A	B	$focus_A(B)$	$focus_B(A)$	Conclusion
comp.sys.mac.apps	comp.sys.mac.system	0.80	0.77	λ-equivalent
rec.crafts.textiles.sewing	comp.lang.perl.misc	0.35	0.32	λ-disjoint
rec.boats.paddle	mixed11	0.88	0.57	λ-subset
rec.sport.volleyball	rec.sport.cricket	0.47	0.46	undetermined

labelling training data for learning the classifier probes in the first place. Additionally, if new categories are added or old categories removed from the hierarchy, new probes must be learned and each source re-probed.

Previous research on grouping terms (as in our source-biased probing with focal terms) has tended to focus on finding terms that are effective for query-expansion [28, 22] or finding lexically similar terms [25]. In both cases, the goal is to find close semantic relationships between terms (e.g. to discover that "Microsoft" and "Windows" are semantically similar). In contrast, our focal term groups are much more coarse-grained, and we make no assertion as to the closeness of the semantic relationship among the terms.

8. CONCLUSIONS

In this paper, we have presented a novel approach for discovering relevant data intensive web services that is based on a service-centric view of the Web. Using source-biased probing allows us to answer service-level queries about the relative size, scope, and relevance of one service to another. In addition, we have discussed how a biased focus measure can serve as a meaningful benchmark of relevance. We have discussed an enhancement to source-biased probing called source-biased probing with focal terms. Our efforts have also extended the basic relevance algorithm to a service neighborhood graph for inferring interesting semantic relationships among multiple services. We are interested in exploring further avenues for the automatic crawling and identification of data intensive web services with which to seed our source-biased probing approach.

9. REFERENCES

[1] J. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *SIGMOD '99*.

[2] J. P. Callan and M. E. Connell. Query-based sampling of text databases. *Information Systems*, 19(2):97–130, 2001.

[3] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks . In *SIGIR '95*, Seattle, WA, 1995.

[4] W. W. Cohen and Y. Singer. Learning to query the web. In *AAAI Workshop on Internet-Based Information Systems*. 1996.

[5] CompletePlanet.com. http://www.completeplanet.com.

[6] N. Craswell, P. Bailey, and D. Hawking. Server selection on the World Wide Web. In *ACM Digital Libraries*, 2000.

[7] R. Dolin, D. Agrawal, and A. Abbadi. Scalable collection summarization and selection. In *ACM Digital Libraries*, 1999.

[8] J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *SIGIR '99*.

[9] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM TOIS*, 17(3):229–229, 1999.

[10] L. Gravano and H. García-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *VLDB '95*.

[11] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: text-source discovery over the Internet. *ACM TODS*, 24(2):229–264, 1999.

[12] D. Hawking and P. Thistlewaite. Methods for information server selection. *ACM TOIS*, 17(1):40–76, 1999.

[13] InvisibleWeb.com. http://www.invisibleweb.com.

[14] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *VLDB '02*, August 2002.

[15] P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *SIGMOD '01*, May 2001.

[16] P. G. Ipeirotis, L. Gravano, and M. Sahami. QProber: A system for automatic classification of hidden-web databases. *ACM TOIS*, 21(1):1–41, 2003.

[17] L. Liu. Query routing in large-scale digital library systems. In *ICDE '99*.

[18] W. Meng, K.-L. Liu, C. T. Yu, X. Wang, Y. Chang, and N. Rishe. Determining text databases to search in the internet. In *VLDB '98*.

[19] W. Meng, C. T. Yu, and K.-L. Liu. Detection of heterogeneities in a multiple text database environment. In *CoopIS '99*.

[20] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[21] A. L. Powell, J. C. French, J. P. Callan, M. E. Connell, and C. L. Viles. The impact of database selection on distributed searching. In *SIGIR '00*.

[22] Y. Qiu and H.-P. Frei. Concept-based query expansion. In *SIGIR '93*.

[23] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *Readings in Information Retrieval*. Morgan Kauffman, San Francisco, CA, 1997.

[24] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *CACM*, 18(11):613–620, 1971.

[25] H. Schutze and J. O. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3):307–318, 1997.

[26] A. Sugiura and O. Etzioni. Query routing for web search engines: Architecture and experiments. In *WWW '00*.

[27] W. Wang, W. Meng, and C. Yu. Concept hierarchy based text database categorization in a metasearch engine environment. In *WISE '00*.

[28] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR '96*.

[29] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the internet. In *Database Systems for Advanced Applications*, 1997.