IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

Efficient Search Engine Measurements

Ziv Bar-Yossef, Maxim Gurevich

CCIT Report #743
August 2009

Electronics
Computers
Communications

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

CCIT REPORT #743 August 2009

Efficient Search Engine Measurements*

Ziv Bar-Yossef † Maxim Gurevich ‡

August 30, 2009

Abstract

We address the problem of externally measuring aggregate functions over documents indexed by search engines, like corpus size, index freshness, and density of duplicates in the corpus. The recently proposed estimators for such quantities [5, 8] are biased due to inaccurate approximation of the so called "document degrees". In addition, the estimators in [5] are quite costly, due to their reliance on rejection sampling.

We present new estimators that are able to overcome the bias introduced by approximate degrees. Our estimators are based on a careful implementation of an approximate importance sampling procedure. Comprehensive theoretical and empirical analysis of the estimators demonstrates that they have essentially no bias even in situations where document degrees are poorly approximated.

By avoiding the costly rejection sampling approach, our new importance sampling estimators are significantly more efficient than the estimators proposed in [5]. Furthermore, building on an idea from [8], we discuss *Rao-Blackwellization* as a generic method for reducing variance in search engine estimators. We show that Rao-Blackwellizing our estimators results in performance improvements, while not compromising accuracy.

1 Introduction

Background. In this paper we focus on external methods for measuring aggregate functions over search engine corpora. These methods interact only with the public interfaces of search engines and do not rely on privileged access to internal search engine data or on specific knowledge of how the search engines work.

One application of our techniques is external evaluation of global quality metrics of search engines. Such evaluation provides the means for objective benchmarking of search engines. Such benchmarks can be used by search engine users and clients to gauge the quality of the service they get

*A preliminary version of this paper appeared in the proceedings of the 16th International World-Wide Web Conference (WWW2007) [3]. Supported by the Israel Science Foundation.

†Google Haifa Engineering Center, Israel, and Dept. of Electrical Engineering, Technion, Haifa 32000, Israel. Email: zivby@ee.technion.ac.il.

‡Dept. of Electrical Engineering, Technion, Haifa 32000, Israel. Email: gmax@tx.technion.ac.il. Supported by the Eshkol Fellowship of the Israeli Ministry of Science.

and by researchers to compare search engines. Even search engines themselves may benefit from external benchmarks, as they can help them reveal their strengths and weaknesses relative to their competitors. Assuming that search engines index a large fraction of the useful web pages, one can use our techniques to study also properties of the web.

Our study concentrates on measurement of global metrics of search engines, like corpus size, index freshness, and density of spam or duplicate pages in the corpus. Such metrics are relevance neutral, and therefore no human judgment is required for computing them. Still, as external access to search engine data is highly restricted, designing automatic methods for measuring these metrics is very challenging. Our objective is to design measurement algorithms that are both *accurate* and *efficient*. Efficiency is particularly important for two reasons. First, efficient algorithms can be executed even by parties whose resources are limited, like researchers. Second, as search engines are highly dynamic, efficient algorithms are necessary for capturing instantaneous snapshots of the search engines.

One might wonder why we study measurement of global index metrics, rather than focusing on the "important" pages that are actually served to users as search results (see a relevant discussion about this in [25]). The latter is indeed a worthy goal, but it requires access to the search engine's query log, which is not publicly available. We argue that global index metrics are useful too, because they provide insight into the search engine's crawling and indexing architecture, which is typically less sensitive to the constantly-changing user query stream. Furthermore, while usage-based metrics are good for evaluating the quality of the search engine relative to the more popular queries, global metrics are better at demonstrating how well the search engine copes with long-tail queries that cannot be predicted a priori.

Problem statement. Let $\mathcal{D}$ denote the corpus of documents indexed by the search engine being measured. We focus on measurement of quantities that can be expressed as either *sums* or *averages* over $\mathcal{D}$. Given a *target function* $f : \mathcal{D} \to \mathbb{R}$, the *sum* of f and the *average* of f are:

$$\mathrm{sum}(f) \ \triangleq \ \sum_{\mathrm{x} \in \mathcal{D}} f(\mathrm{x}), \quad \mathrm{avg}(f) \ \triangleq \ \frac{\mathrm{sum}(f)}{|\mathcal{D}|}.$$

(In fact, we address sums and averages w.r.t. arbitrary measures, not just the uniform one. See more details in Section 3. For simplicity of exposition, in the introduction we focus on the uniform measure.) Almost all search engine metrics we are aware of can be expressed as sum or average of some function. For example, the corpus size, $|\mathcal{D}|$, is the sum of the constant 1 function ($f(\mathrm{x}) = 1$ for all x); the density of spam pages in the corpus is the average of the spam indicator function ($f(\mathrm{x}) = 1$, if x is a spam page, and $f(\mathrm{x}) = 0$, otherwise); the number of unique documents in the corpus is the sum of the inverse duplicate-count function ($f(\mathrm{x}) = 1/d_{\mathrm{x}}$, where d_{x} is the number of duplicates x has, including x itself). Many other metrics, like search engine overlap, sizes of subsets of the corpus, or index freshness can be expressed as sums or averages as well. We allow also sums and averages of vector-valued functions $f : \mathcal{D} \to \mathbb{R}^m$, which can be used to compute histograms of the indexed pages (e.g., by language, country domain, or topic).

A *search engine estimator* for $\mathrm{sum}(f)$ (resp., $\mathrm{avg}(f)$) is a probabilistic procedure, which submits queries to the search engine, fetches pages from the web, computes the target function f on documents of its choice, and eventually outputs an estimate of $\mathrm{sum}(f)$ (resp., $\mathrm{avg}(f)$). The quality of an estimator is measured in terms of its *bias* and its *variance*. The efficiency of an estimator is measured in terms of the number of queries it submits to the search engine, the number of web pages it fetches, and the number of documents on which it computes the target function f.

State of the art. Brute-force computation of functions over a search engine corpus is infeasible, due to the huge size of the corpus and the highly restricted access to it. Every user is limited to a few thousand queries per day and only the top k matches are returned. k is typically at most $1,000$ and may vary from query to query depending on the search engine architecture, the current load on it, etc. We are not aware of a technique for accessing all the results of a query.

An alternative to brute-force computation is sampling. One samples random uniform pages from the corpus and uses them to estimate the desired quantity. If the samples are unbiased, then a small number of them is sufficient to obtain accurate estimations. The main challenge is to design algorithms that can efficiently generate uniform unbiased samples from the corpus using queries to the public interface. Bharat and Broder [6] were the first to propose such an algorithm. The samples produced by their algorithm, however, suffered from severe bias towards long, content-rich, documents. In our previous paper [5], we were able to correct this bias by proposing a technique for *simulating* unbiased sampling by biased sampling. To this end, we applied several stochastic simulation methods, like rejection sampling [30] and the Metropolis-Hastings algorithm [26, 16]. Stochastic simulation, however, incurs significant overhead: in order to generate each unbiased uniform sample, numerous biased samples are used, and this translates into elevated query and fetch costs. For instance, our most efficient sampler needed about 2,000 queries to generate each uniform sample.

In an attempt to address this lack of efficiency, we also experimented [5] with *importance sampling* estimation. Importance sampling [24, 19] enables estimation of sums and averages *directly* from the biased samples, without first generating unbiased samples. This technique can significantly reduce the stochastic simulation overhead. Nevertheless, our estimators in [5] used stochastic simulation twice (once to select random queries and once to select random documents), and we were able to use importance sampling to eliminate only the latter of the two. Furthermore, our importance sampling estimator was still wasteful, as it used only a single result of each submitted query and discarded all the rest.

Broder *et al.* [8] have recently made remarkable progress by proposing a new estimator for search engine corpus size. Their estimator (implicitly) employs importance sampling. Moreover, the estimator somehow makes use of *all* query results in the estimation and is thus less wasteful than the estimators in [5]. Broder *et al.* claimed their method can be generalized to estimate other metrics, but have not provided any details.

The degree mismatch problem. A prerequisite for applying importance sampling is the ability to compute for each biased sample an *importance weight*. The importance weights are used to balance the contributions of the different biased samples to the final estimator.

In the estimators of [5, 8], computing the importance weight of a sample document translates into calculation of the document's "degree". Given a large pool of queries (e.g., "phrase queries of length 5", or "8-digit string queries"), the *degree* of a document w.r.t. the pool is the number of queries from the pool to whose results x belongs. As the estimators of [5, 8] choose their sample documents from the results of random queries drawn from a query pool, these samples are biased towards high degree documents. Document degrees, therefore, constitute the primary factor in determining the importance weights of sample documents.

As importance weights (and hence degrees) are computed for every sample document, degree computation should be extremely efficient. Ideally, it should be done based on the content of x alone and without submitting queries to the search engine. The above estimators do this by extracting all the terms/phrases from x and counting how many of them belong to the pool. The resulting number is the document's *predicted degree* and is used as an approximation of the real degree.

In practice, the predicted degree may be quite different from the actual degree, since we do not exactly know how the search engine parses documents and extracts terms from them or how it selects the terms by which to index the document. Moreover, the document may fail to belong to the result sets of some queries it matches if it is ranked too low (beyond the top k results) or if search engine did not return it because of high load. These factors give rise to a *degree mismatch*—a gap between the predicted degree and the actual degree. The degree mismatch implies that the importance weights used by the estimators are not accurate, and this can significantly affect the quality of the produced estimates.

In [5], we proved that if the density of overflowing (having more than k results) queries among all the queries that a document matches has low variance, then the bias incurred by degree mismatch is small. However, in that work we did not analyze additional factors causing degree mismatch. Broder *et al.* [8] have not analyzed the effect of degree mismatch on the quality of their estimations at all.

Several heuristic methods have been used by [5, 8] to overcome the degree mismatch problem. In order to reduce the effect of overflowing queries, a pool of queries that are unlikely to overflow was chosen ([5] used a pool of phrases of length 5, while [8] used a pool of 8-digit strings). [8] remove potentially overflowing queries from the pool by eliminating terms that occur frequently in a training corpus. However, this heuristic can have many false positives or false negatives, depending on the choice of the frequency threshold.

Our contributions. In this paper we show how to overcome the degree mismatch problem. We present four search engine estimators that remain nearly unbiased and efficient, even in the presence of highly mismatching degrees.

Our starting point (Section 4) is a new importance sampling estimator for search engine metrics, which generalizes the corpus size estimator of Broder *et al.* [8]. Our previous estimator [5] worked in two steps: first it sampled queries from some query pool (using rejection sampling) and then it sampled documents from the results of these queries (using either rejection sampling or importance sampling). The new estimator, on the other hand, samples queries and documents *together* in a single step, using importance sampling. Avoiding the costly rejection sampling step makes this estimator significantly more efficient than the estimators in [5].

We then analyze (Section 4.6) the effect of the degree mismatch problem on the accuracy of this importance sampling estimator (the analysis applies also to the previous estimators [5, 8]). We prove that the estimator suffers from significant bias that depends on how far the approximate degrees are from the real degrees. Since the dependence is multiplicative, even slightly skewed degrees may result in significant estimation bias. Indeed, our empirical study (Section 10.2) reveals that the Broder *et al.* corpus size estimator suffers from relative bias of about 7400%, due to the effect of degree mismatch.

The algorithms used in [5, 8] to compute document degrees are deterministic and base their calculations only on the content of the document whose degree is being computed. These algorithms are very efficient, yet, as we show in Section 5, are not accurate. We present a new algorithm for estimating document degrees. The algorithm is probabilistic and needs to send the search engine a small number of queries in order to produce an estimate. However, the output of this algorithm is a provably unbiased estimate of the real document degree. By plugging in the estimated degrees into the importance sampling estimator, we obtain SumEst, a provably unbiased estimator for sum metrics (Section 7)[1].

Average metrics are more tricky to handle, because for such metrics the target measure is known only up to normalization. For example, when the target measure is the uniform distribution, the normalization constant $|\mathcal{D}|$ is not known in advance. As a result, the bias of the importance sampling estimator depends not only on how well degrees are approximated, but also on the unknown normalization constant.

The standard approach for dealing with unknown normalization constants is to apply *ratio importance sampling* (cf. [23]), which divides the normal importance sampling estimator by an estimate of the normalization constant. We present a new variant of ratio importance sampling, which we call *approximate ratio importance sampling* (or, *ARIS*, for short), which neutralizes the effect of both the unknown normalization constant and the approximate degrees. We use ARIS and the probabilistic degree estimator to obtain AvgEst—a search engine estimator for average metrics (Section 8). This estimator is nearly unbiased (the bias diminishes to 0 with the number of samples).

SumEst and AvgEst resort to the probabilistic degree estimator, which is more accurate but also more costly than the deterministic degree estimators used in previous works. In Section 9, we show how to replace the probabilistic degree estimators in SumEst and AvgEst by the efficient deterministic degree estimator, while not compromising the estimation accuracy significantly. The resulting estimators, which we call EffSumEst and EffAvgEst, are considerably more efficient than SumEst and AvgEst.

Our last contribution builds on the observation that the estimator of Broder *et al.* implicitly applies *Rao-Blackwellization* [9], which is a well-known statistical tool for reducing estimation variance. This technique is what makes their estimator so efficient. Since Rao-Blackwellization increases the number of degree computations, it cannot be efficiently applied to SumEst and AvgEst, where degree computations require submitting search engine queries. We thus apply Rao-Blackwellization to EffSumEst and EffAvgEst estimators only and prove that it is guaranteed to make them more efficient as long as the results of queries are sufficiently variable.

We emphasize that our estimators are applicable to both sum and average metrics. This in contrast to the estimators in [5], which are efficiently applicable only to average metrics, and the estimator in [8], which is applicable only to sum metrics.

Experimental results. We evaluated the bias and the efficiency of our estimators as well as the estimators from [5, 8] on a local search engine that we built over a corpus of 2.4 million documents.

[1] The estimator is unbiased relative to the documents covered by the query pool used. Like all other estimators that use query pools, the estimator may have bias due to uncovered documents. See more details in Section 3.5.

To this end, we used the estimators to estimate two different metrics: corpus size and density of sports pages. The empirical study confirms our analytical findings: in the presence of significant degree mismatch, our estimators have essentially no bias, while the estimator of Broder *et al.* suffers from significant bias. For example, the relative bias of SumEst in the corpus size estimation was 0.01%, while the relative bias of the estimator of Broder *et al.* was 60%. The study also showed that our new estimators are up to 1500 times more efficient than the rejection sampling estimator from [5]. Finally, the study demonstrated the effectiveness of Rao-Blackwellization by reducing the query cost of estimators by up to 80%.

We used our estimators to measure the absolute sizes of two major search engines. The results of this study may underestimate the true search engine sizes, largely due to the limited coverage of search engine corpora by the pool of queries we used. Even so, we showed that our estimates are up to 3.5 times higher than the estimates produced by (our implementation of) the *Broder et al.* estimator.

2 Related work

Apart from [6, 5, 8], several other studies estimated global metrics of search engine indices, like relative corpus size. These studies are based on analyzing anecdotal queries [7], queries collected from user query logs [20, 13], or queries selected randomly from a pool a la Bharat and Broder [15, 10]. Using capture-recapture techniques (cf. [22]) some of these studies infer measurements of the whole web. Due to the bias in the samples, though, these estimates lack any statistical guarantees.

A different approach for evaluating search quality is by sampling pages from the whole web [21, 17, 18, 2, 27]. Sampling from the whole web, however, is a more difficult problem, and therefore all the known algorithms suffer from severe bias.

Anagnostopoulos, Broder, and Carmel [1] showed a technique for measuring parameters of the results of a single search engine query, rather of the entire corpus. Their technique, however, assumes privileged access to the internal data structures of the search engine.

Dasgupta, Das, and Mannila [11] presented a random walk algorithm for sampling records from a database that is hidden behind a web form. A search engine is essentially an example of such database. However, as this work is aimed at structured database setting, it is not directly applicable to sampling from a free text search engine.

In our subsequent work [4] we proposed a technique for sampling real user queries via query suggestion services of search engines. Such query samples can then be used to measure metrics as *observed* by the users, e.g., the observed diversity of search results, amount of spam, etc. We also showed that using these query samples, instead of a synthetic query pool, one can sample web pages proportionally to their ImpressionRank, i.e., their actual visibility to the search engine users.

In order to avoid disturbing the flow of the paper, some proofs are postponed to the appendix.

Notation	Meaning
$\mathcal{D}$	Set of documents indexed by a search engine.
$\mathcal{Q}$	The query space of a search engine.
Ω	Generic finite space.
π	Generic target measure on Ω.
ρ^n	The distribution induced by a measure ρ.
Z_ρ	The normalization constant of a measure ρ.
$\mathrm{sum}_\pi(f)$	Sum of f relative to the target measure π.
$\mathrm{avg}_\pi(f)$	Average of f relative to the target measure π.
$\pi_{\mathcal{D}}$	Target measure on $\mathcal{D}$.
G	A generic queries-documents graph.
$\mathcal{P}$	Query pool.
$\mathsf{documents}_G(q)$	The set of documents incident to a query q in G.
$\mathsf{queries}_G(\mathrm{x})$	The set of queries incident to a document d in G.
$\deg_G$	The vertex degree function in G.

Table 1: Notation used in Section 3.

3 Framework for search engine measurements

In this section we introduce notations and definitions used to formally describe our search engine estimators.

3.1 Search engines

Definition 3.1 (Search engine)**.** *A* search engine *is a 4-tuple* $\langle \mathcal{D}, \mathcal{Q}, \mathsf{results}(\cdot), k \rangle$*, where:*

1. *$\mathcal{D}$ is the document corpus indexed. Documents are assumed to have been pre-processed (e.g., they may be truncated to some maximum size limit).*
2. *$\mathcal{Q}$ is the query space supported by the search engine. A query is a sequence of one or more terms.*
3. *$\mathsf{results}(\cdot)$ is a mapping that maps every query $q \in \mathcal{Q}$ to an ordered sequence of documents, called* results*. Whenever a user sends a query q as a query to the search engine, the search engine returns $\mathsf{results}(q)$ to the user.*
4. *k is the* result set size limit *(typically $k = 1,000$). The k most highly ranked query results are returned, i.e., $\mathsf{results}(q) \leq k$ for each q. The actual number of results may be lower than k and may vary from query to query depending on the search engine architecture, the current load on it, etc.*

We stress that $\mathsf{results}(q)$ are the *actual* results returned by the search engine on q. For many queries there are more matches than the ones that are actually returned. Typical search engines (e.g., Google, Bing, Yahoo!) return the $1,000$ most highly ranked results. For example, in Google, $\mathsf{results}$("britney spears") consists of only 747 results, while the total number of matches reported by Google on this query is $77,000,000$.

3.2 Search engine estimators

We are interested in measurement of quantities that can be written as sums of functions over a finite space:

$$\mathrm{sum}_{\pi}(f) \triangleq \sum_{\mathrm{x}\in\Omega} f(\mathrm{x})\pi(\mathrm{x}).$$

Here, Ω is a finite space, $f : \Omega \to \mathbb{R}$ is a *target function* and $\pi : \Omega \to [0, \infty)$ is a *target measure*.

The average of a function is essentially a sum where the target measure is a probability distribution. We say that measure ρ *induces* a corresponding probability distribution on Ω:

$$\rho^n(\mathrm{x}) = \frac{\rho(\mathrm{x})}{Z_\rho},$$

where $Z_\rho = \sum_{\mathrm{x}\in\Omega} \rho(\mathrm{x})$ is the *normalization constant* of ρ. We say that two different measures are the same *up to normalization*, if they induce the same probability distribution, but have different normalization constants.

The average of a function is then reduced to a sum as follows:

$$\mathrm{avg}_{\pi}(f) \triangleq \mathrm{sum}_{\pi^n}(f).$$

For example, suppose Ω is the set of all documents indexed by a search engine. Then, the corpus size of the search engine is a sum of the function $f(\mathrm{x}) = 1, \forall \mathrm{x}$ relative to the uniform target measure ($\pi(\mathrm{x}) = 1$ for all $\mathrm{x} \in \Omega$). The density of spam pages in the corpus is an average of the function $f(\mathrm{x}) = 1$ iff x is a spam page relative to the same uniform target measure. Alternatively, it is a sum of the same function relative to the uniform distribution on Ω ($\pi^n(\mathrm{x}) = 1/|\Omega|$ for all x).

Everything we do in this paper can be generalized to deal with vector-valued functions $f : \Omega \to \mathbb{R}^m$. Yet, for simplicity of exposition, we focus on scalar functions.

Definition 3.2 (Search engine estimator)**.** *Let $\pi_{\mathcal{D}}$ be a target measure on a document corpus $\mathcal{D}$ indexed by a search engine. A* search engine estimator *for* $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$ *(resp.,* $\mathrm{avg}_{\pi_{\mathcal{D}}}(f)$*) is a randomized procedure P, that is given access to four "oracle" procedures:*

1. `computeTargetMeasure`(x)*: returns the weight $\pi_{\mathcal{D}}(x)$ of a document $x \in \mathcal{D}$ relative to the target measure $\pi_{\mathcal{D}}$.*
2. `getResults`(q)*: returns the results returned by the search engine on a query q.*
3. `getDocument`(x)*: returns the HTTP header and the content of a document x.*
4. `computeFunction`(x)*: returns the value of function f on a document x.*

Each invocation of the estimator produces an estimate of $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$ *(resp.,* $\mathrm{avg}_{\pi_{\mathcal{D}}}(f)$*). Different invocations of the estimator produce identically distributed and independent outputs.*

Estimation quality. The quality of an estimator M is measured in terms of its *bias* and *variance*. The *bias* of M is $\mathrm{bias}(\mathrm{M}) = \mathbb{E}(\mathrm{M}) - \mathrm{sum}_{\pi_{\mathcal{D}}}(f)$ and its *variance* is $\mathrm{var}(\mathrm{M}) = \mathbb{E}((\mathrm{M} - \mathbb{E}(\mathrm{M}))^2)$. M is called *unbiased*, if $\mathrm{bias}(\mathrm{M}) = 0$.

If the variance of an estimator is high, it may have high error, even if it has low bias. To circumvent this problem, estimators are typically designed in two steps. First, we design a *basic estimator* that has low bias and possibly high variance. Then, to reduce the variance, we create a final estimator by aggregating multiple independent instances of the basic estimator. The simplest aggregation method is averaging: $\mathrm{T} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{M}_i$, where $\mathrm{M}_1, \ldots, \mathrm{M}_n$ are n independent instances of the basic estimator M. T has the same bias as the basic estimator, but its variance tends to 0 as n tends to infinity. By Chebyshev's inequality, $n = O\left(\frac{\mathrm{var}(\mathrm{M})}{\epsilon^2 \cdot \mathbb{E}^2(\mathrm{M})}\right)$ independent instances of M are sufficient to guarantee that the estimate produced by M falls within the confidence interval $(1 \pm \epsilon) \cdot \mathbb{E}(\mathrm{M})$ with constant confidence (e.g., 2/3).[2]

Estimation costs. The three expensive resources used by search engine estimators are: (1) queries submitted to the search engine; (2) web pages fetched; (3) calculations of the function f. Queries and web page fetches may take substantial amount of time and require usage of network bandwidth. Queries are costly also because search engines pose daily quotas on the number of queries they are willing to accept from a single user.[3] Depending on the function f, each calculation of f may require substantial processing time, fetching web pages, or submitting more queries to the search engine. We therefore use three measures of efficiency for search engine estimators: *query cost*, *fetch cost*, and *function cost*.

The *expected query cost* of an estimator M, denoted qcost(M), is the expected number of queries M submits to the search engine. Note that in order to compare the efficiency of different estimators by their expected query cost, the variance of both estimators should be the same. The *amortized query cost*, defined as $\mathrm{qcost}(\mathrm{M}) \cdot \frac{\mathrm{var}(\mathrm{M})}{E^2(\mathrm{M})}$, is a more robust measure of efficiency. By Chebyshev's inequality, amortized query cost determines the number of queries required to obtain an estimate within a given confidence interval.

Expected and amortized fetch/function costs are defined similarly.

3.3 Queries-documents graph

A "queries-documents graph" $G = (\mathcal{P}, \mathcal{D}, \mathcal{E})$ is a bipartite graph, whose left side is a *query pool* $\mathcal{P} \subseteq \mathcal{Q}$ and whose right side is the document corpus $\mathcal{D}$. All the estimators we consider in this paper sample documents from $\mathcal{D}$ by working over a properly defined queries-documents graph: they sample queries from the query pool $\mathcal{P}$ and then sample document neighbors of these queries in the graph. In Section 5 we define three types of queries-documents graphs used in this paper.

[2] Somewhat more efficient aggregation techniques, like the *median of averages* (cf. [14]), exist. For simplicity of exposition, we will focus mainly on averaging in this paper.

[3] The daily quotas apply to the developer APIs provided by search engines. Queries sent to the standard web interfaces are also rate limited.

We denote by $\mathsf{documents}_G(q)$ all the documents in $\mathcal{D}$ incident to q in G (i.e., $\{\mathrm{x}|(q, \mathrm{x}) \in \mathcal{E}\}$). The *degree* of a query q in G is then $\deg_G(q) = |\mathsf{documents}_G(q)|$. The degree of a set of queries $\mathcal{P}' \subseteq \mathcal{P}$ is defined as: $\deg_G(\mathcal{P}') = \sum_{q \in \mathcal{P}'} \deg_G(q)$.

Similarly, by $\mathsf{queries}_G(\mathrm{x})$ we denote all the queries in $\mathcal{P}$ incident to x in G (i.e., $\{q|(q, \mathrm{x}) \in \mathcal{E}\}$). The *degree* of x in G is $\deg_G(\mathrm{x}) = |\mathsf{queries}_G(\mathrm{x})|$. The degree of a set of documents $\mathcal{D}' \subseteq \mathcal{D}$ is defined as: $\deg_G(\mathcal{D}') = \sum_{\mathrm{x} \in \mathcal{D}'} \deg_G(\mathrm{x})$.

3.4 Incidence computation

The estimators we use in this paper require "local accessibility" to the queries-documents graph G. By that we mean that the estimator needs efficient implementations of the following procedures that compute incidences in the graph:

1. $\mathtt{getDocuments}_G(q)$: Given a query $q \in \mathcal{P}$, returns all documents that are incident to q in G, i.e., $\mathsf{documents}_G(q)$.
2. $\mathtt{getDegree}_G(\mathrm{x})$: Given a document $\mathrm{x} \in \mathcal{D}$, returns the number of queries that are incident to x in G, i.e., $\deg_G(\mathrm{x})$.

Note that our algorithms do not need to compute $\mathsf{queries}_G(\mathrm{x})$, but rather $\deg_G(\mathrm{x})$ only. As we show later, this allows more efficient implementation, since we can estimate $\deg_G(\mathrm{x})$ at lower cost than computing $\mathsf{queries}_G(\mathrm{x})$. We describe the implementation of these procedures in Section 5.

3.5 Corpus coverage

Some documents in $\mathcal{D}$ may be "isolated"—they have no edges incident to them in the queries-documents graph G. This means the estimators that rely on G base their estimates only on the non-isolated documents in $\mathcal{D}$, i.e., $\mathcal{D}_G \triangleq \{\mathrm{x}| \deg_G(\mathrm{x}) > 0\}$. Clearly, we would like then $\mathcal{D}_G$ to "cover" as much of $\mathcal{D}$ as possible.

We denote the *coverage* of $\mathcal{D}$ by G relative to the target measure $\pi_{\mathcal{D}}$ as $\pi^n_{\mathcal{D}}(\mathcal{D}_G)$, where $\pi^n_{\mathcal{D}}$ is the distribution induced by $\pi_{\mathcal{D}}$. For instance, when $\pi^n_{\mathcal{D}}$ is the uniform distribution, the coverage of $\mathcal{D}$ by G is $|\mathcal{D}_G|/|\mathcal{D}|$. For the above reasons, we would like the coverage to be as close to 1 as possible. Note that even if the coverage is lower than 1, but $\mathcal{D}_G$ is sufficiently representative of $\mathcal{D}$, then estimators that use G produce accurate estimates of parameters of the whole corpus $\mathcal{D}$.

To simplify the presentation, we assume from now on that $\mathcal{D}_G = \mathcal{D}$. The reader should keep in mind that the estimators we present may incur additional bias, in case the coverage of $\mathcal{D}$ is low.

3.6 Assumptions

Like the pool-based estimators in [6, 5, 8], our estimator assumes knowledge of an *explicit* query pool $\mathcal{P}$. For example, in our experiments, we used a pool of 2.37 billion numeric and English terms and two-term conjunctions. Such a pool can be constructed in a preprocessing step, by crawling a representative corpus of web documents and extracting terms that occur therein (we used Wikipedia and the ODP [12] directory for this purpose). We can run the estimator with any such pool, yet the choice of the pool may affect the bias and the efficiency of the estimator. See Section 6.2 for an explanation how the properties of the pool affect the quality of our estimators.

Our estimators have indirect and highly restricted access to search engines' indices. To guarantee the correctness of our algorithms in such settings, we need to make the following assumptions:

Dynamic corpora. Our algorithms assume that search engine corpora do not change during the estimation process. Obviously, this assumption does not hold in practice as search engine indices are constantly being updated. In our experiments we noticed only slight differences in the results returned for the same queries at different steps of the experiment. We note that the duration of our experiments was determined by the limited resources we used. Having more resources could have shortened this duration and drastically diminished the effect of corpus changes.

Versioned indices. Search engines may maintain multiple non-identical versions of the index simultaneously and serve users from different versions of the index (based on the user's profile or based on load-balancing criteria). Our algorithms assume all queries are served from a single coherent index. If all the queries are indeed served from the same version of the index, then the results produced by our algorithms reflect properties of the specific index version used. Some anomalies may occur, if the samplers work with multiple index versions simultaneously, assuming the differences among the versions are significant (which we do not believe to be the case in most search engines).

4 Search engine importance sampling estimation

Notation	**Meaning**
$\mathrm{supp}(\rho)$	Support of a measure ρ.
IS	The importance sampling estimator.
w	The importance weight function.
π	Extension of the target measure $\pi_{\mathcal{D}}$ to $\mathcal{P} \times \mathcal{D}$.
$\mathcal{P}_G$	The queries in $\mathcal{P}$ that have at least one incident document in G.
p	Trial distribution over $\mathcal{P} \times \mathcal{D}$.
(Q, X)	A random query-document pair distributed according to p.
PSE	The pool size estimator.
AIS	The approximate importance sampling estimator.

Table 2: Notation used in Section 4.

In this section we present a basic importance sampling search engine estimator. It will be used as a basis for the more accurate and efficient estimators presented in subsequent sections.

4.1 The naive estimator

The naive Monte Carlo estimator (cf. [23]) for $\text{avg}_\pi(f)$ works as follows: (1) generate a random sample X from the distribution π^n induced by π; (2) output $f(\mathrm{X})$. It is easy to check that this estimator is unbiased.

In our setting, however, this simple estimator is inefficient, for the following reasons: (1) sampling from the distribution π^n may be hard or costly; (e.g., when π^n is a uniform distribution on Ω); (2) the random variable $f(\mathrm{X})$ may have high variance.

Moreover, the naive estimator is not suitable for computing sums. Although a sum of f can be computed as $Z_\pi \cdot \text{avg}_\pi(f)$, Z_π cannot be computed using the naive estimator, requiring computing it by other means, which may be hard or costly (e.g., in corpus size estimation, Z_π is the corpus size, which is exactly the quantity we need to estimate).

We will use importance sampling to obtain an efficient estimator that is also able to estimate sums.

4.2 The importance sampling estimator

The basic idea of importance sampling [24, 19, 23] is the following. Instead of generating a sample Y from the target distribution π^n, the estimator generates a sample X from a different *trial distribution* p on $\mathcal{D}$. p can be any distribution, as long as $\text{supp}(p) \supseteq \text{supp}(\pi)$ (here, $\text{supp}(p) = \{\mathrm{x} \in \mathcal{D} \mid p(\mathrm{x}) > 0\}$ is the *support* of p; $\text{supp}(\pi)$ is defined similarly). In particular, we can choose it to be a distribution that is easy to sample from. The importance sampling estimator is then defined as follows:

$$\mathrm{IS}(\mathrm{X}) \;\triangleq\; f(\mathrm{X}) \cdot \frac{\pi(\mathrm{X})}{p(\mathrm{X})} \;=\; f(\mathrm{X}) \cdot w(\mathrm{X}).$$

The correction term

$$w(\mathrm{x}) \triangleq \frac{\pi(\mathrm{x})}{p(\mathrm{x})},$$

where $\mathrm{x} \in \mathcal{D}$, is called the "importance weight".

Theorem 4.1. $\mathrm{IS}(X)$ *is an unbiased estimator for* $\text{sum}_\pi(f)$.

Proof.

$$\mathbb{E}(\mathrm{IS}(\mathrm{X})) \;=\; \sum_{\mathrm{x} \in \text{supp}(p)} p(\mathrm{x}) f(\mathrm{x}) w(\mathrm{x}) \;=\; \sum_{\mathrm{x} \in \text{supp}(p)} p(\mathrm{x}) f(\mathrm{x}) \frac{\pi(\mathrm{x})}{p(\mathrm{x})} \;=\; \sum_{\mathrm{x} \in \text{supp}(\pi)} f(\mathrm{x}) \pi(\mathrm{x}) \;=\; \text{sum}_\pi(f).$$

□

In fact, we do not have to compute importance weights exactly, it is enough to have an unbiased estimator $W(\mathrm{x})$, such that $\mathbb{E}(W(\mathrm{x})) = w(\mathrm{x})$:

$$\mathbb{E}(\mathrm{IS}(\mathrm{X})) \;=\; \sum_{\mathrm{x} \in \text{supp}(p)} p(\mathrm{x}) f(\mathrm{x})\, \mathbb{E}(W(\mathrm{x})) \;=\; \sum_{\mathrm{x} \in \text{supp}(p)} p(\mathrm{x}) f(\mathrm{x}) w(\mathrm{x}) \;=\; \text{sum}_\pi(f).$$

The variance of the importance sampling estimator depends on the variance of the product $f(\mathrm{Y}) \cdot w(\mathrm{Y})$:

$$\mathrm{var}(\mathrm{IS}(\mathrm{X})) = \mathrm{var}(f(\mathrm{X}) \cdot w(\mathrm{X})).$$

Observe that the variance is minimized when $f(\mathrm{X})$ and $w(\mathrm{X}) = \pi(\mathrm{X})/p(\mathrm{X})$ are anti-correlated. Typically, the trial distribution p is selected to be correlated with $f(\mathrm{X}) \cdot \pi(\mathrm{X})$, leading to low estimation variance. In our case however, we have little freedom in choosing the trial distribution, as it depends on the available data provided by the search engine through its public interface. Thus, we use a predefined trial distribution and measure the resulting estimation variance empirically.

Implementation of an importance sampling estimator requires: (1) ability to sample efficiently from the trial distribution p; and (2) ability to compute the importance weight $w(\mathrm{x})$ (or its estimator $W(\mathrm{x})$) and the function value $f(\mathrm{x})$, for any given element $\mathrm{x} \in \mathcal{D}$. There is no need to know the normalization constant Z_π or to be able to sample from π^n. This basic importance sampling estimator is only suitable for estimating sums. We later (Section 8) extend it for estimating averages.

4.3 The sample space

The sample space of the importance sampling estimator proposed in our previous paper [5] was the corpus of documents $\mathcal{D}$. The trial distribution p was the "document degree distribution", in which documents are sampled proportionally to their degrees in the queries-documents graph induced by $\mathcal{P}$. In order to sample documents from this distribution, we had to sample queries from the pool $\mathcal{P}$ proportionally to their degrees (referred to as "cardinalities" in our previous paper), and then to sample random documents from the result sets of these queries. As degrees of queries are not known in advance, sampling queries from $\mathcal{P}$ required application of rejection sampling. This step incurred significant overhead.

In this paper we propose a different sample space. Rather than sampling queries and then documents in two separate steps, we sample them *together*. Let $G = (\mathcal{P}, \mathcal{D}, \mathcal{E})$ be a queries-documents graph induced by a pool $\mathcal{P}$ (we define $\mathcal{E}$ later in Section 5). The sample space we use is $\mathcal{E} \subseteq \mathcal{P} \times \mathcal{D}$, and each sample is a query-document pair (q, x). We extend the target measure $\pi_{\mathcal{D}}$ on $\mathcal{D}$ into a target measure π on $\mathcal{E}$, and function f on $\mathcal{D}$ into a function F on $\mathcal{E}$. The extension is done in such a way that $\mathrm{sum}_\pi(F)$ equals $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$. We thus reduce the problem of estimating $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$ to the problem of estimating $\mathrm{sum}_\pi(F)$. For the latter, we can apply importance sampling directly on the two-dimensional sample space $\mathcal{E}$, without having to resort to rejection sampling as an intermediate step.

We extend the function f on $\mathcal{D}$ into a function F on $\mathcal{E}$ as follows: $F(q, \mathrm{x}) \triangleq f(\mathrm{x})$. Similarly, we extend $\pi_{\mathcal{D}}$ into a measure on $\mathcal{E}$ as follows:

$$\pi(q, \mathrm{x}) \triangleq \frac{I(\mathrm{x} \in \mathsf{documents}_G(q)) \cdot \pi_{\mathcal{D}}(\mathrm{x})}{\deg_G(\mathrm{x})},$$

where I is an indicator function: $I(\text{condition}) = 1$ if the condition is true, and 0 otherwise. The marginal weight of a document x relative to this measure equals its weight relative to the measure $\pi_{\mathcal{D}}$; this weight splits evenly among edges incident to x. The connection between π and $\pi_{\mathcal{D}}$ is given by the following proposition:

Proposition 4.2. *$\pi_{\mathcal{D}}$ is the marginal measure of π on $\mathcal{D}$. Furthermore, the normalization constants of $\pi_{\mathcal{D}}$ and π are the same.*

Proof. To prove that $\pi_{\mathcal{D}}$ is the marginal measure of π, we must show that for every $\mathrm{x} \in \mathcal{D}$, $\pi_{\mathcal{D}}(\mathrm{x}) = \sum_{q \in \mathcal{P}} \pi(q, \mathrm{x})$:

$$\begin{aligned}
\sum_{q \in \mathcal{P}} \pi(q, \mathrm{x}) &= \sum_{q \in \mathcal{P}} \frac{I(\mathrm{x} \in \mathsf{documents}_G(q)) \cdot \pi_{\mathcal{D}}(\mathrm{x})}{\deg_G(\mathrm{x})} \\
&= \frac{\pi_{\mathcal{D}}(\mathrm{x})}{\deg_G(\mathrm{x})} \cdot \sum_{q \in \mathsf{queries}_G(\mathrm{x})} 1 \\
&= \frac{\pi_{\mathcal{D}}(\mathrm{x})}{\deg_G(\mathrm{x})} \cdot \deg_G(\mathrm{x}) \\
&= \pi_{\mathcal{D}}(\mathrm{x}).
\end{aligned}$$

Now, the identity of the two normalization constants easily follows:

$$Z_\pi = \sum_{\mathrm{x} \in \mathcal{D}} \sum_{q \in \mathcal{P}} \pi(q, \mathrm{x}) = \sum_{\mathrm{x} \in \mathcal{D}} \pi_{\mathcal{D}}(\mathrm{x}) = Z_{\pi_{\mathcal{D}}}.$$

□

It follows from the proposition that π is a distribution if and only if $\pi_{\mathcal{D}}$ is a distribution.

Proposition 4.3. $\mathrm{sum}_\pi(F) = \mathrm{sum}_{\pi_{\mathcal{D}}}(f)$

Proof.

$$\mathrm{sum}_\pi(F) = \sum_{q \in \mathcal{P}, \mathrm{x} \in \mathcal{D}} \pi(q, \mathrm{x})\ F(q, \mathrm{x}) = \sum_{\mathrm{x} \in \mathcal{D}} f(\mathrm{x}) \sum_{q \in \mathcal{P}} \pi(q, \mathrm{x}) = \sum_{\mathrm{x} \in \mathcal{D}} f(\mathrm{x}) \pi_{\mathcal{D}}(\mathrm{x}) = \mathrm{sum}_{\pi_{\mathcal{D}}}(f).$$

□

4.4 The trial distribution

We next describe the trial distribution for sampling edges from $\mathcal{E}$. Let $\mathcal{P}_G$ denote the collection of queries in $\mathcal{P}$ that have at least one incident document in G:

$$\mathcal{P}_G \triangleq \{q \in \mathcal{P} \mid \mathsf{documents}_G(q) \neq \emptyset\}.$$

Our trial distribution selects an edge (q, x) as follows: (1) pick a query $q \in \mathcal{P}_G$ uniformly at random; (2) pick a document $\mathrm{x} \in \mathsf{documents}_G(q)$ uniformly at random:

$$p(q, \mathrm{x}) \triangleq \frac{1}{|\mathcal{P}_G|} \cdot \frac{I(\mathrm{x} \in \mathsf{documents}_G(q))}{\deg_G(q)}.$$

Sampling from p can be done easily (see Function 1): we repeatedly select queries from $\mathcal{P}$ uniformly at random and call $\texttt{getDocuments}_G$ to get their incident documents (by submitting these queries to the search engine; detailed implementation in Section 5). We stop when reaching a query that has at least one incident document. We then select a document from the set of incident documents of this query uniformly at random.

Function 1 `samplePair`(G)

while true **do**
  Q := uniformly chosen query from $\mathcal{P}$
  $\mathsf{documents}_G(\mathrm{Q}) := \mathtt{getDocuments}_G(\mathrm{Q})$
  if $\mathsf{documents}_G(\mathrm{Q}) \neq \emptyset$ **then**
    X := uniformly chosen document from $\mathsf{documents}_G(\mathrm{Q})$
    return (Q, X)

4.5 The importance sampling search engine estimator

The importance weights corresponding to the target measure π and the trial distribution p are the following:

$$w(q, \mathrm{x}) \;=\; \frac{\pi(q, \mathrm{x})}{p(q, \mathrm{x})} \;=\; \frac{\pi_{\mathcal{D}}(\mathrm{x}) \cdot |\mathcal{P}_G| \cdot \deg_G(q)}{\deg_G(\mathrm{x})}.$$

Thus, the importance sampling estimator for $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$ is:

$$\mathrm{IS}(\mathrm{Q}, \mathrm{X}) \;\triangleq\; \frac{f(\mathrm{X}) \cdot \pi_{\mathcal{D}}(\mathrm{X}) \cdot |\mathcal{P}_G| \cdot \deg_G(\mathrm{Q})}{\deg_G(\mathrm{X})},$$

where (Q, X) is a sample from the trial distribution p.

The only term in the $\mathrm{IS}(\mathrm{Q}, \mathrm{X})$ we do not yet know how to compute is $|\mathcal{P}_G|$. As exactly computing it is infeasible, due to a large size of $\mathcal{P}$, we resort to probabilistic estimation of $|\mathcal{P}_G|$ by the "Pool Size Estimator" PSE (see Function 2). If we sample a query Q uniformly at random from $\mathcal{P}$, it has a probability of $\frac{|\mathcal{P}_G|}{|\mathcal{P}|}$ to have at least one incident document. Therefore, we can estimate $\frac{|\mathcal{P}_G|}{|\mathcal{P}|}$ as follows: repeatedly sample queries uniformly at random from $\mathcal{P}$ and compute its degree; the fraction of submitted queries that have non-zero degree is an unbiased estimator for $\frac{|\mathcal{P}_G|}{|\mathcal{P}|}$. Estimation error can be reduced by increasing the number of iterations performed (n).

Function 2 `estimatePoolSize`(G)

$j := 0$
for $i := 1$ to n **do**
  Q := uniformly chosen query from $\mathcal{P}$
  $\deg_G(\mathrm{Q}) := |\mathtt{getDocuments}_G(\mathrm{Q})|$
  if $\deg_G(\mathrm{Q}) > 0$ **then**
    $j := j + 1$
return $|\mathcal{P}| \cdot j/n$

Function 3 is the implementation of the IS search engine estimator.

4.6 Importance sampling with approximate degrees

Unfortunately, due to the degree mismatch problem (see detailed description in Section 5), we are unable to accurately compute document degrees, and consequently the importance weight function

Function 3 $\texttt{IS}(G)$

$\text{PSE} := \texttt{estimatePoolSize}(G)$
$(\text{Q}, \text{X}) := \texttt{samplePair}(G)$
$\pi_{\mathcal{D}}(\text{X}) := \texttt{computeTargetMeasure}(\text{X})$
$\deg_G(\text{Q}) := |\texttt{getDocuments}_G(\text{Q})|$
$\deg_G(\text{X}) := \texttt{getDegree}_G(\text{X})$
$w(\text{Q}, \text{X}) := \pi_{\mathcal{D}}(\text{X}) \cdot \text{PSE} \cdot \deg_G(\text{Q}) / \deg_G(\text{X})$
return $\texttt{computeFunction}(\text{X}) \cdot w(\text{Q}, \text{X})$

w. We now analyze the effect of the approximate importance weights on the importance sampling estimator.

An *approximate importance sampling* estimator employs an "approximate importance weight function" $u(\text{x})$ rather than the exact one $w(\text{x})$. We prove that the estimation generated by approximate importance sampling is close to the true value as long as the importance weight function $w(\text{x})$ and the approximate importance weight function $u(\text{x})$ are similar. Here we extend our result from previous work [5] which considered the special case of "approximate trial weights". To the best of our knowledge, no previous study addressed this scenario before and it could be of independent interest.

Let $u(\text{x})$ be an approximate weight function. Define the *approximate importance sampling estimator* as:

$$\text{AIS}(\text{X}) = f(\text{X}) \cdot u(\text{X}),$$

where X is distributed according to the trial distribution p. Suppose $\text{supp}(u) \subseteq \text{supp}(w)$. The following lemma analyzes the bias of the approximate importance sampling estimator:

Theorem 4.4.

$$\mathbb{E}(\text{AIS}(X)) \;=\; \text{sum}_\pi(f) \cdot \mathbb{E}\left(\frac{u(Y)}{w(Y)}\right) + Z_\pi \cdot \text{cov}\left(f(Y), \frac{u(Y)}{w(Y)}\right),$$

where $X \propto p$, $Y \propto \pi^n$, and Z_π is the normalization constant of π.

For the proof, see Appendix A.

It follows from the theorem that there are two sources of bias in this estimator: (1) multiplicative bias, depending on the expectation of u/w relative to π^n; and (2) additive bias, depending on the correlation between f and u/w and on the normalization constant Z_π. Note that the multiplicative factor, even if small, may have a significant effect on the estimator's bias, and thus must be eliminated. The additive bias is typically less significant, as in many practical situations f and u/w are uncorrelated (e.g., when f is a constant function as is the case with corpus size estimation).

The estimators we present in Sections 7 and 8 use two alternative strategies for eliminating the multiplicative bias in the approximate importance sampling estimator. The former employs AIS with probabilistic approximate importance weights that are unbiased estimates of the corresponding real importance weights, and consequently the multiplicative bias is 1. The latter estimates the multiplicative bias incurred by AIS and then divides $AIS(\text{X})$ by this estimate, effectively neutralizing the multiplicative bias.

5 Incidence and degree computation

Notation	Meaning
S	The search queries graph.
P	The predicted queries graph.
V	The valid queries graph.
vdensity(x)	Validity density on document x.

Table 3: Notation used in Section 5.

In this section we analyze the causes of the degree mismatch problem. We demonstrate that computation of incidences and degrees in the search queries graph is tricky, and simple workarounds, like the ones used in previous works, give rise to degree mismatch. We then propose a new probabilistic algorithm for estimating document degrees, which is provably unbiased.

5.1 The search queries graph

In the search queries graph, $S = (\mathcal{P}, \mathcal{D}, \mathcal{E}_S)$, a query $q \in \mathcal{P}$ is connected to a document $\mathrm{x} \in \mathcal{D}$ if and only if the search engine returns x as a result on query q (i.e., $\mathrm{x} \in \mathsf{results}(q)$). Thus, the efficient implementation of the procedure $\mathtt{getDocuments}_S(q)$, (see Function 4), is trivial: just submit q to the search engine and output all the results returned. The cost of this implementation is a single search engine query.

Function 4 $\mathtt{getDocuments}_S(q)$

1: submit q to the search engine
2: $\mathsf{documents}(q) :=$ results returned by the search engine
3: **return** $\mathsf{documents}(q)$

Remark. An "overflowing query" is a query that has more matching documents than what the search engine actually returns on that query. Every query that has more than k (typically, $k = 1,000$) matches is overflowing, because search engines return up to k results for each query. Some queries with fewer than k matches can be overflowing too (for technical reasons, search engines sometimes do not return all the matches they have for a query).

In our previous papers [5, 3], our estimators did not use overflowing queries from the query pool. We found this restriction to be unnecessary. Therefore, the search queries graph consists of both overflowing and non-overflowing queries. An overflowing query α is connected only to the documents that the search engine actually returns on α, not to the other matching documents that it does not return.

Unfortunately, we do not know how to efficiently implement the second procedure, $\mathtt{getDegree}_S(\mathrm{x})$. It can be implemented by submitting each query from $\mathcal{P}$ to the search engine and returning those that have x as their result. This implementation is impractical, due to the large number of queries in a typical query pool.

5.2 The predicted queries graph

We now describe a different queries-documents graph, admitting a more efficient method for degree computation.

Let $P = (\mathcal{P}, \mathcal{D}, \mathcal{E}_P)$ be the *predicted queries* graph, where the neighbors of a particular document x are defined based on x's content alone. Let the *predicted queries* of x, $\mathsf{queries}_P(\mathrm{x})$, be the set of queries that occur in the content of x. For example, if $\mathcal{P}$ is the pool of single term queries, $\mathsf{queries}_P(\mathrm{x})$ is the set of all distinct terms that occur in the text of x and that also occur in $\mathcal{P}$. If $\mathcal{P}$ is the pool of two-term conjunctions, $\mathsf{queries}_P(\mathrm{x})$ is the set of all pairs of distinct terms that occur in the text of x and that also occur in $\mathcal{P}$. In Function 5 we show how to compute $\mathsf{queries}_P(\mathrm{x})$ using a single page fetch and without submitting any queries to the search engine.[4]

Function 5 $\mathtt{getQueries}_P(\mathrm{x})$

1: download x
2: $\mathsf{queries}_P(\mathrm{x}) :=$ queries in $\mathcal{P}$ that the content of x matches
3: **return** $\mathsf{queries}_P(\mathrm{x})$

$\deg_P(\mathrm{x})$ is the degree of x in the predicted queries graph, i.e., $|\mathsf{queries}_P(\mathrm{x})|$. It can be efficiently computed using a single document fetch (see Function 6).

Function 6 $\mathtt{getDegree}_P(\mathrm{x})$

1: $\mathsf{queries}_P(\mathrm{x}) := \mathtt{getQueries}_P(\mathrm{x})$
2: **return** $|\mathsf{queries}_P(\mathrm{x})|$

Unfortunately, in the predicted queries graph the implementation of $\mathtt{getDocuments}_P(q)$ becomes difficult. In order to find the documents that are incident to a query q, we need to go over all documents in $\mathcal{D}$ and find the ones in which q occurs. This amounts to full-fledged indexing of the whole corpus $\mathcal{D}$, which is clearly infeasible for the setting we consider in this paper. Note that we cannot use the search engine itself to implement $\mathtt{getDocuments}_P(q)$, at least directly, because it returns $\mathsf{documents}_S(q)$, which may be different from $\mathsf{documents}_P(q)$ (see details below).

5.3 Combining the search queries and the predicted queries graphs

We just saw how to efficiently compute query incidences in the search queries graph and document incidences in the predicted queries graph. Can we somehow combine the two to obtain efficient implementations of both?

The straightforward solution, used in previous works [8, 5], is to simply call $\mathtt{getDocuments}_S(q)$ whenever we need to compute query incidences and to call $\mathtt{getDegree}_P(\mathrm{x})$ whenever we need to compute document degrees. The assumption here is that $\deg_P(\mathrm{x})$ is a good approximation of $\deg_S(\mathrm{x})$. We show below that this assumption is frequently false and may lead to what we call the "degree mismatch problem". This mismatch results in significant estimation error, as we demonstrate in Section 10.

[4]We assume an efficient data structure for checking membership in $\mathcal{P}$.

We distinguish between two types of error when approximating $\mathsf{queries}_S(\mathrm{x})$ by $\mathsf{queries}_P(\mathrm{x})$: (1) *false negatives*: $\mathsf{queries}_P(\mathrm{x})$ may miss some queries that belong to $\mathsf{queries}_S(\mathrm{x})$; (2) *false positives*: $\mathsf{queries}_P(\mathrm{x})$ may contain queries that do not belong to $\mathsf{queries}_S(\mathrm{x})$.

In the following we list several factors that cause false negatives:

1. Indexing depth. We assumed the first d (where d is some constant; d was set to $20,000$ in our experiments) terms in each document are indexed. If the search engine's indexing depth is greater than d, terms/phrases that occur beyond the d-th position in x will not be included in $\mathsf{queries}_P(\mathrm{x})$, although the search engine may return x as one of the results.

2. Parsing and tokenization. Different search engines may have slightly different algorithms for parsing and tokenizing documents. For example, two words separated by a comma may or may not be indexed as a phrase. If the search engine's parser determines a sequence of characters in x to be a term or a sequence of terms in x to be a phrase, while our parser does not, the corresponding term/phrase will not be included in $\mathsf{queries}_P(\mathrm{x})$, although the search engine may return x as one of the results.

3. Indexing by terms not appearing in document's content. Search engines index documents under terms that do not occur at their text at all (e.g., anchor text terms or synonyms). Our procedure for computing $\mathsf{queries}_P(\mathrm{x})$, obviously, will not find a document to match such terms (unless they appear in the document's content too).

False positives are caused by the following factors:

1. Overflowing queries. As mentioned above, search engines do not always return all the matches to queries they receive. If x has low rank, it may not be returned on overflowing queries that it matches.

2. Duplicates and near-duplicates. If the search engine filters duplicate or near-duplicate documents, a query that x matches may not return x as a result, if one of the documents that are similar to x is returned as a result.

3. Host collapsing. If the search engine collapses documents belonging to the same host, a query that matches x may not return x as a result, if another document from the same host is returned as a result.

4. Indexing depth. If the search engine's indexing depth is smaller than d, the search engine may not return x as a result on terms that occur beyond its indexing depth.

5. Parsing and tokenization. If our parser determines a sequence of characters in x to be a term or a sequence of terms in x to be a phrase, while the search engine's parser does not, the search engine may not return x as a result on the corresponding term/phrase.

5.4 The valid queries graph

How can we bridge the gap between $\mathsf{queries}_S(\mathrm{x})$ and $\mathsf{queries}_P(\mathrm{x})$? We define a *valid* queries-documents graph $V = (\mathcal{P}, \mathcal{D}, \mathcal{E}_V)$, where $\mathcal{E}_V = \mathcal{E}_S \bigcap \mathcal{E}_P$. That is, an edge (q, x) exists in the valid queries graph if and only if it exists in both the search queries and the predicted queries graphs. That is, the set of queries incident to a document x is:

$$\mathsf{queries}_V(\mathrm{x}) \triangleq \mathsf{queries}_S(\mathrm{x}) \cap \mathsf{queries}_P(\mathrm{x}),$$

and the set of documents incident to a query q is:

$$\mathsf{documents}_V(q) \triangleq \mathsf{documents}_S(q) \cap \mathsf{documents}_P(q).$$

Some of our algorithms use the valid queries graph, rather than the search queries or the predicted queries graphs.

In order to work with the valid queries graph, we need to clarify three points: (1) can we efficiently implement the procedure $\texttt{getDocuments}_V(q)$? (2) can we efficiently implement $\texttt{getDegree}_V(\mathrm{x})$? and (3) what is the coverage of V?

We start with the first question. Note that $\mathsf{documents}_V(q)$ can be rewritten as follows:

$$\mathsf{documents}_V(q) = \{\mathrm{x} \in \mathsf{documents}_S(q) \mid q \in \mathsf{queries}_P(\mathrm{x})\}.$$

Implementing $\texttt{getDocuments}_V(q)$ is then straightforward (see Function 7), since we know how to efficiently compute $\mathsf{documents}_S(q)$ and $\mathsf{queries}_P(\mathrm{x})$. The procedure first gets q's incident documents in the search queries graph $\mathsf{documents}_S(q)$ (see Function 4). Then, it fetches all these documents, and checks, for each document x, whether $q \in \texttt{getQueries}_P(\mathrm{x})$. It then returns the set of documents $\mathsf{documents}_V(q)$. The cost of this implementation is a single search engine query, and $|\mathsf{documents}_S(q)|$ document fetches.

Function 7 $\texttt{getDocuments}_V(q)$

$\mathsf{documents}_V(q) := \emptyset$
$\mathsf{documents}_S(q) := \texttt{getDocuments}_S(q)$
for all $\mathrm{x} \in \mathsf{documents}_S(q)$ **do**
  download x
  $\mathsf{queries}_P(\mathrm{x}) := \texttt{getQueries}_P(\mathrm{x})$
  if $q \in \mathsf{queries}_P(\mathrm{x})$ **then**
    $\mathsf{documents}_V(q) := \mathsf{documents}_V(q) \bigcup \{\mathrm{x}\}$
return $\mathsf{documents}_V(q)$

We address the second question about degree calculation in the next subsection. As for the third question, we note that V's coverage may indeed decrease compared to the search queries graph. If all the edges incident to a document x in S are not present in P, then this document will become isolated in the valid queries graph. This could happen, for example, with documents that have little or no text content and are indexed mainly by anchor text terms. It is left for future work to study the reduction in the graphs's coverage due to a smaller number of edges in V compared to S.

5.5 Computing document degrees in the valid queries graph

We are left to show how to compute document degrees in the valid queries graph. We next present three alternative techniques.

Naive approximation. We can simply use $\deg_P(\mathrm{x})$ as an approximation of $\deg_V(\mathrm{x})$ (see Function 6). The advantage of this technique is its low cost: no queries to the search engine are submitted. Its disadvantage, as shown in Section 5.3, is its imprecision. We know that $\deg_P(\mathrm{x}) \geq \deg_V(\mathrm{x})$, so we always overestimate document degrees.

The quality of this approximation depends on the precision within which the predicted queries graph approximates the search queries graph. Formally, the *validity density* of a document x is defined as the fraction of x's predicted queries that are also search queries:

$$\mathrm{vdensity}(\mathrm{x}) \;=\; \frac{|\mathsf{queries}_P(\mathrm{x}) \bigcap \mathsf{queries}_S(\mathrm{x})|}{\mathsf{queries}_P(\mathrm{x})} \;=\; \frac{\deg_V(\mathrm{x})}{\deg_P(\mathrm{x})}.$$

From the above, $\mathrm{vdensity}(\mathrm{x}) \in [0, 1]$ for all x. The closer it is to 1, the better is the naive approximation of $\deg_V(\mathrm{x})$ by $\deg_P(\mathrm{x})$.

Brute force calculation. $\deg_V(\mathrm{x})$ can be calculated by submitting each query q from $\mathsf{queries}_P(\mathrm{x})$ to the search engine and returning the number of queries for which $\mathrm{x} \in \mathsf{documents}_S(q)$ (see Function 8).

Function 8 $\mathtt{getDegree}_V(\mathrm{x})$

$\mathsf{queries}_P(\mathrm{x}) := \mathtt{getQueries}_P(\mathrm{x})$
$i := 0$
for all $q \in \mathsf{queries}_P(\mathrm{x})$ **do**
  $\mathsf{documents}_S(q) := \mathtt{getDocuments}_S(q)$
  if $\mathrm{x} \in \mathsf{documents}_S(q)$ **then**
    $i := i + 1$
return i

The advantage of this technique is its accuracy: it returns the exact degree in V. Its disadvantage is its cost: it requires submitting all the queries in $\mathsf{queries}_P(\mathrm{x})$ (up to thousands of queries per document) to the search engine—a prohibitively expensive task.

Sampling-based approximation. Here we combine the above two techniques to create a reasonably accurate, yet practical method for computing degrees. We start by computing $\mathsf{queries}_P(\mathrm{x})$. We then randomly select from $\mathsf{queries}_P(\mathrm{x})$ a small subset of r queries and submit each of the selected queries to the search engine. Finally, we probabilistically estimate vdensity(x) by the fraction α of queries (q) in the sample for which $\mathrm{x} \in \mathsf{documents}_S(q)$. An unbiased estimator for $\deg_V(\mathrm{x})$ is then $\alpha \cdot \deg_P(\mathrm{x})$ (see Function 9).

As we show later, we will need to calculate $1/\deg_V(\mathrm{x})$ rather than $\deg_V(\mathrm{x})$. Unfortunately, the inverse of the above estimator yields a *biased* estimator for $1/\deg_V(\mathrm{x})$. We thus resort to directly estimating $1/\deg_V(\mathrm{x})$ by the "Inverse Degree Estimator" (IDE(x)) described below.

Function 10 shows a procedure for estimating $1/\deg_V(\mathrm{x})$ for a given document x, using a limited number of queries. The procedure repeatedly samples queries uniformly at random from the set of predicted queries $\mathsf{queries}_P(\mathrm{x})$. It submits each query to the search engine and checks whether they are in $\mathsf{documents}_S(\mathrm{Q})$. The procedure stops when reaching the first query that is in $\mathsf{documents}_S(\mathrm{Q})$. The number of queries sampled so far is geometrically distributed with vdensity(x) as the success

Function 9 $\texttt{estimateDegree}_V(\mathrm{x})$

```
queries_P(x) := getQueries_P(x)
i := 1
for j := 1 to r do
    Q := uniformly chosen query from queries_P(x)
    documents_S(q) := getDocuments_S(Q)
    if x ∈ documents_S(Q) then
        i := i + 1
return (i/r) · |queries_P(x)|
```

parameter. The expectation of this estimator is exactly $\frac{1}{\mathrm{vdensity(x)}} / \deg_P(\mathrm{x}) = 1/\deg_V(\mathrm{x})$. Note that the procedure is always guaranteed to terminate, because we apply it only on documents x for which $\deg_V(\mathrm{x}) > 0$.

Function 10 $\texttt{estimateInverseDegree}_V(\mathrm{x})$

```
queries_P(x) := getQueries_P(x)
i := 1
while true do
    Q := uniformly chosen query from queries_P(x)
    documents_S(q) := getDocuments_S(Q)
    if x ∈ documents_S(Q) then
        return i/|queries_P(x)|
    i := i + 1
```

6 Overview of the estimators

In the following sections we combine the tools developed in the previous sections to design search engine estimators for sums and for averages. For sums, the estimator SumEst (described in Section 7) is a straightforward application of the importance sampling estimator on the valid queries graph V. For averages, we apply, in Section 8, *approximate importance sampling* to design AvgEst. Finally, we show in Section 9 how to apply Rao-Blackwellization to obtain more efficient versions of both estimators: EffSumEst and EffAvgEst.

6.1 Summary

Tables 4 and 5 summarize the bias and efficiency guarantees of the search engine estimators we propose. Empirical analysis of the estimation variance is given in the experimental results (Section 10).

Table 4 shows that SumEst has no bias at all, while the bias of AvgEst diminishes to 0 with the number of iterations n performed by the estimator. EffSumEst and EffAvgEst may have higher bias, depending on the correlation between the target function f and the expected validity density

Estimator	Objective	Bias
SumEst	sum	0
AvgEst	average	$O(1/n)$
EffSumEst	sum	$Z_{\pi_{\mathcal{D}}} \cdot \frac{\mathrm{cov}(f(\mathrm{Y}),\mathrm{vdensity}(\mathrm{Y}))}{\mathbb{E}(\mathrm{vdensity}(\mathrm{Y}))} + O\left(\frac{1}{n}\right) \quad (\mathrm{Y} \propto \pi_{\mathcal{D}}^n)$
EffAvgEst	average	$\frac{\mathrm{cov}(f(\mathrm{Y}),\mathrm{vdensity}(\mathrm{Y}))}{\mathbb{E}(\mathrm{vdensity}(\mathrm{Y}))} + O\left(\frac{1}{n}\right) \quad (\mathrm{Y} \propto \pi_{\mathcal{D}}^n)$

Table 4: Worst-case guarantees on the bias of the search engine estimators, after n iterations.

Estimator	Query Cost	Fetch Cost	Function Cost
SumEst, AvgEst	$n \cdot \left(\frac{\lvert\mathcal{P}\rvert}{\lvert\mathcal{P}_V\rvert} + \mathbb{E}\left(\frac{1}{\mathrm{vdensity}(\mathrm{X})}\right)\right)$ $(\mathrm{X} \propto p_{\mathcal{D}})$	$n \cdot \frac{\lvert\mathcal{P}\rvert}{\lvert\mathcal{P}_V\rvert} \cdot \mathrm{avg}_{q \in \mathcal{P}} \deg_S(q)$	n
EffSumEst, EffAvgEst	$n \cdot \frac{\lvert\mathcal{P}\rvert}{\lvert\mathcal{P}_V\rvert}$	$n \cdot \frac{\lvert\mathcal{P}\rvert}{\lvert\mathcal{P}_V\rvert} \cdot \mathrm{avg}_{q \in \mathcal{P}} \deg_S(q)$	$n \cdot \mathrm{avg}_{q \in \mathcal{P}_V} \deg_V(q)$

Table 5: Expected costs of n iterations of the search engine estimators.

of documents. Note that for some target functions, like the constant function $f \triangleq 1$ used for corpus size estimation, this correlation is 0 irrespective of the document validity density.

The costs of SumEst and AvgEst (see Table 5) are combinations of four components: (1) The number of iterations n performed in order to reduce estimation bias and/or variance; (2) The ratio between the total number of queries in $\mathcal{P}$ and the number of queries in $\mathcal{P}_V$; (3) The expected inverse document validity density; and (4) The average degree of queries in S and in V. The gap between the query cost of SumEst and AvgEst and the query cost of EffSumEst and EffAvgEst is $n \cdot \mathbb{E}\left(\frac{1}{\mathrm{vdensity}(\mathrm{X})}\right)$, which is the larger of the two terms that comprise the query cost of SumEst and AvgEst.

The fetch costs of all estimators are the same. The function cost of the EffSumEst and EffAvgEst is higher than the function cost of SumEst and AvgEst by a factor of $\mathrm{avg}_{q \in \mathcal{P}_V} \deg_V(q)$.

The tables demonstrate the superiority of SumEst and AvgEst in terms of bias and the advantage of EffSumEst and EffAvgEst in terms of cost. The bias of EffSumEst and EffAvgEst estimators remains low as long as the target function is not correlated with document validity density.

6.2 Considerations in choosing the query pool and the predicted queries graph

Recall (Section 5.4) that the valid queries graph (V) used by our estimators is an intersection of the search queries graph (S) and the predicted queries graph (P). In order for these estimators to be accurate and efficient, we need V to be as close as a possible to S. Since the document corpus (the right hand side of each of these graphs) and the edges of the search graph S are dependent on the search engine, we do not have any control over their choice. We do have freedom in choosing the query pool $\mathcal{P}$ (the left hand side of each of the above graphs) and the edges of the predicted queries graph. We overview below considerations that should be taken into account when making these choices.

Corpus coverage. As mentioned in Section 3.5, the estimator's bias depends on the coverage of $\mathcal{D}$ by V. In order to achieve high coverage by V, we need: (a) to achieve high coverage of $\mathcal{D}$ by S; and (b) make sure that V contains enough edges from S, so that the coverage of $\mathcal{D}$ by V is similar to the coverage of $\mathcal{D}$ by S.

The coverage of $\mathcal{D}$ by S depends only on the choice of the query pool $\mathcal{P}$. The pool must consist of sufficiently many queries so that almost every document in $\mathcal{D}$ is actually returned by the search engine on some query(s) in $\mathcal{P}$. The main difficulty is in covering low PageRank documents that are usually not returned on popular queries. To address such documents, the pool must consist of many "long-tail" queries that have few matching results (fewer than the result set size limit k). The search engine is more likely to return low PageRank documents on such queries. We found phrase queries, consisting of multiple terms, to be a good source of such long-tail queries. Another example is "inurl:" queries, that return only pages whose URL matches query keywords. In our empirical study, we used inurl queries of long numerical strings and conjunctions of two terms extracted from Wikipedia and from the ODP corpus.

In order to make sure P has enough edges from S, we would like as many of the search queries incident to each document x in $\mathcal{D}$ to be also incident to x in the predicted queries graph. If we had access to the index of the search engine, we could have connected x in P to all the queries by which x is indexed. Note that this set of queries is a superset of the queries that are incident to x in S, because the latter are the queries on which the search engine returns x as one of the top k results. As we do not have access to the index of the search engine, all we can do is try to mimic the indexing process of the search engine. That is, we need to find the sources of queries by which the document is indexed (e.g., the document's URL, content, anchor text), to parse these sources into tokens, to determine phrase boundaries, etc. In our empirical study we used only the document's URL as the source for predicted queries and we tried to mimic the tokenization techniques used by major search engines.

Validity density. The query cost of SumEst and AvgEst depends on the document validity density: the higher the expected validity density, the lower the query cost of SumEst and AvgEst. Also the bias of EffSumEst and EffAvgEst depends on validity density: if the validity density is high (close to 1), its variance is low, and consequently also its correlation with any target function is low. Thus, high validity density is beneficial for reducing the bias of EffSumEst and EffAvgEst.

How do we promote high validity density? Recall (Section 5.3) that a major factor causing low validity density is overflowing queries (queries on which not all matching documents are returned). One way to dilute the effect of overflowing queries is to use many "long-tail" queries in the pool, where each such query has a small number of matching documents, but collectively they cover sufficiently many documents from the corpus.

Pool size. Recall that $\mathcal{P}_V$ denotes the collection of queries in $\mathcal{P}$ having at least one incident document in the valid queries graph. The query and the fetch costs increase as $\frac{|\mathcal{P}_V|}{|\mathcal{P}|}$ decreases. One should thus avoid pools including queries likely to return little or no results, such as pools consisting of very long phrase queries or of conjunctive queries comprising of many unrelated terms.

Average query degree. The fetch and the function costs depend on the average degree of queries in S and in V, respectively. We would thus like the average query degrees in these graphs to be as low as possible. Of course, this may interfere with the coverage: if the average degree is too low, more queries are needed to achieve sufficient coverage.

Result set size limit k. Higher k typically leads to higher variance of query degrees. This higher variance, in turn, increases the variance of the importance weights and, consequently, the variance of the importance sampling estimators. Thus, the amortized costs increase with k.

7 Importance sampling estimator for sums

Notation	Meaning
SumEst	The estimator for sums.
w_{sum}	The importance weight function of the estimator for sums.
u	The approximate importance weight function of the estimators for sums and averages.
$p_{\mathcal{D}}$	The marginal distribution of p on $\mathcal{D}$.

Table 6: Notation used in Section 7.

Recall that in order to employ the importance sampling estimator (see Section 4.5), we need to compute the importance weight function

$$w_{\mathrm{sum}}(q, \mathrm{x}) \;=\; \frac{\pi_{\mathcal{D}}(\mathrm{x}) \cdot |\mathcal{P}_V| \cdot \deg_V(q)}{\deg_V(\mathrm{x})},$$

where (q, x) is a sample from the trial distribution p.

We use the estimation procedures described in Section 5 (Function 2 for estimating $|\mathcal{P}_V|$, Function 7 for computing $\deg_V(q)$, and Function 10 for estimating $1/\deg_V(\mathrm{x})$), applied to the valid queries graph, to compute the following estimate of the importance weight function:

$$u(q, \mathrm{x}) \;=\; \pi_{\mathcal{D}}(\mathrm{x}) \cdot \mathrm{PSE} \cdot \deg_V(q) \cdot \mathrm{IDE}(\mathrm{x}).$$

(Recall that PSE is the estimator for $|\mathcal{P}_V|$ and IDE(x) is the estimator for $1/\deg_V(\mathrm{x})$.) The following proposition is based on the fact that PSE and IDE(x) are independent estimators, so that $\mathbb{E}(\mathrm{PSE} \cdot \mathrm{IDE}(\mathrm{x})) = \mathbb{E}(\mathrm{PSE}) \cdot \mathbb{E}(\mathrm{IDE}(\mathrm{x}))$.

Proposition 7.1. *$u(q, x)$ is an unbiased estimator for $w_{sum}(q, x)$.*

Proof.

$$\begin{aligned}
\mathbb{E}(u(q, \mathrm{x})) &= \pi_{\mathcal{D}}(\mathrm{x}) \cdot \deg_V(q) \cdot \mathbb{E}(\mathrm{PSE} \cdot \mathrm{IDE}(\mathrm{x})) \\
&= \pi_{\mathcal{D}}(\mathrm{x}) \cdot \deg_V(q) \cdot |\mathcal{P}_V| \cdot \frac{1}{\deg_V(\mathrm{x})} \\
&= w_{\mathrm{sum}}(q, \mathrm{x}).
\end{aligned}$$

□

Therefore, as shown in Section 4.2, we can use $u(q, \mathrm{x})$ in place of $w_{\mathrm{sum}}(q, \mathrm{x})$. The basic estimator for sums is then defined as follows:

$$f(\mathrm{X}) \cdot u(\mathrm{Q}, \mathrm{X}),$$

where (Q, X) is distributed according to p. Function 11 is the implementation of the full estimator for sums SumEst, defined as follows:

$$\mathrm{SumEst} = \frac{1}{n} \sum_{i=1}^{n} f(\mathrm{X}_i) \cdot u(\mathrm{Q}_i, \mathrm{X}_i).$$

Function 11 `estimateSum`

```
1:  PSE := estimatePoolSize(V)
2:  sum_πD := 0
3:  for i := 1 to n do
4:     (Q, X) := samplePair(V)
5:     π_D(X) := computeTargetMeasure(X)
6:     deg_V(Q) := |getDocuments_V(Q)|
7:     IDE(X) := estimateInverseDegree_V(X)
8:     u(Q, X) := π_D(X) · PSE · deg_V(Q) · IDE(X)
9:     sum_πD := sum_πD + computeFunction(X) · u(Q, X)
10: return sum_πD / n
```

Caching. Naively implementing the above estimator would result in a waste of search engine queries and document fetches, as queries and documents would be submitted and fetched multiple times. For example, the last query submitted by `samplePair`(V) (line 4) is then submitted by `getDocuments`$_V(\mathrm{Q})$ (line 6). To completely avoid resource waste, we employ query and document caching. That is, before we submit a query to the search engine, we first check whether we already have the query's results in the cache. If we do, we avoid submitting a query again. Documents are handled similarly.

7.1 Analysis

By Theorem 4.1:

Proposition 7.2. *SumEst is an unbiased estimator for* $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$.

We now analyze the query cost, the fetch cost, and the function cost of `estimateSum`. The following proposition is used to state the query cost.

Proposition 7.3. *The marginal distribution of* p *on* $\mathcal{D}$ *is:*

$$p_{\mathcal{D}}(x) = \frac{1}{|\mathcal{P}_V|} \sum_{q \in \mathsf{queries}_V(x)} \frac{1}{\deg_V(q)}.$$

Proof. To prove that $p_{\mathcal{D}}(\mathrm{x})$ is the marginal distribution of the trial distribution p, we must show that for every $\mathrm{x} \in \mathcal{D}$, $p_{\mathcal{D}}(\mathrm{x}) = \sum_{q \in \mathcal{P}} p(q, \mathrm{x})$:

$$\sum_{q \in \mathcal{P}} p(q, \mathrm{x}) = \sum_{q \in \mathcal{P}} \frac{1}{|\mathcal{P}_V|} \cdot \frac{I(\mathrm{x} \in \mathsf{documents}_V(q))}{\deg_V(q)} = \frac{1}{|\mathcal{P}_V|} \sum_{q \in \mathsf{queries}_V(\mathrm{x})} \frac{1}{\deg_V(q)}.$$

□

Theorem 7.4.

$$\mathbb{E}(\mathrm{qcost}(\textit{estimateSum})) \;=\; n \cdot \left(\frac{|\mathcal{P}|}{|\mathcal{P}_V|} + \mathbb{E}\left(\frac{1}{\mathrm{vdensity}(X)}\right)\right),$$

where X is distributed according to $p_{\mathcal{D}}$.

$$\mathbb{E}(\mathrm{fetchcost}(\textit{estimateSum})) \;=\; n \cdot \frac{|\mathcal{P}|}{|\mathcal{P}_V|} \cdot \mathrm{avg}_{q \in \mathcal{P}} \deg_S(q).$$

The function cost of ***estimateSum*** *is n.*

Recall that $\mathcal{P}$ is the query pool (see Section 3.3), P is the predicted queries graph (see Section 5.2), V is the valid queries graph (see Section 5.4), and S is the search queries graph (see Section 5.1).

Proof. Search engine queries are submitted in the following functions of the estimator:

- `estimatePoolSize`(V)
- `samplePair`(V)
- `getDocuments`$_V$(Q)
- `estimateInverseDegree`$_V$(X)

We now analyze query cost of each of these procedures.

We assume `estimatePoolSize`(V) is called only once in a preprocessing step and its estimate is used for all subsequent estimator invocations. Therefore we do not include its cost in the cost of `estimateSum` and of the following estimators. It is easy to see that it requires as little as $O(|\mathcal{P}|/|\mathcal{P}_V|)$ queries to produce an accurate estimate.

`samplePair`(V) submits queries, selected uniformly at random from $\mathcal{P}$, until it encounters a query q such that $\mathsf{documents}_V(q) \neq \emptyset$. The query cost of this procedure is then a geometric random variable with success parameter $\frac{|\mathcal{P}_V|}{|\mathcal{P}|}$, whose expectation is

$$\mathbb{E}(\mathrm{qcost}(\texttt{samplePair}(V))) \;=\; \frac{|\mathcal{P}|}{|\mathcal{P}_V|}.$$

`getDocuments`$_V$(Q) submits exactly one search engine query. However, since this query was necessarily already submitted by `samplePair`(V), and since query results are cached, the query cost of `getDocuments`$_V$(Q) is 0.

As shown in Section 5.5, $\texttt{estimateInverseDegree}_V(\mathrm{x})$ submits $\frac{\deg_P(\mathrm{x})}{\deg_V(\mathrm{x})} = \frac{1}{\mathrm{vdensity}(\mathrm{x})}$ queries in expectation. Then, since the random choices of $\texttt{estimateInverseDegree}_V(\mathrm{X})$ are independent of X,

$$\mathbb{E}(\mathrm{qcost}(\texttt{estimateInverseDegree}_V(\mathrm{X}))) = \mathbb{E}\left(\frac{1}{\mathrm{vdensity}(\mathrm{X})}\right).$$

We now analyze the fetch cost. Documents are fetched in the following functions of the estimator:

- $\texttt{samplePair}(V)$
- $\texttt{getDocuments}_V(\mathrm{Q})$
- $\texttt{estimateInverseDegree}_V(\mathrm{X})$

$\texttt{samplePair}(V)$ submits queries, selected uniformly at random from $\mathcal{P}$, until it encounters a query q such that $\mathsf{documents}_V(q) \neq \emptyset$. As we saw earlier, the expected number of queries it probes before returning a query from $\mathcal{P}_V$ is $\frac{|\mathcal{P}|}{|\mathcal{P}_V|}$. For each of these queries, $\texttt{getDocuments}_V(\mathrm{Q})$ is called. $\texttt{getDocuments}_V(\mathrm{Q})$ fetches exactly $\deg_S(\mathrm{Q})$ documents. As Q is chosen from $\mathcal{P}$ uniformly at random, the expected fetch cost of $\texttt{getDocuments}_V(\mathrm{Q})$ is

$$\mathbb{E}(\mathrm{fetchcost}(\texttt{getDocuments}_V(\mathrm{Q}))) \;=\; \frac{1}{|\mathcal{P}|}\sum_{q\in\mathcal{P}} \deg_S(q) = \mathrm{avg}_{q\in\mathcal{P}} \deg_S(q).$$

By Wald's identity (see, e.g., [28], Section 2.2), the expected fetch cost of $\texttt{samplePair}(V)$ is

$$\mathbb{E}(\mathrm{fetchcost}(\texttt{samplePair}(V))) \;=\; \frac{|\mathcal{P}|}{|\mathcal{P}_V|} \cdot \mathrm{avg}_{q\in\mathcal{P}} \deg_S(q).$$

When $\texttt{getDocuments}_V(\mathrm{Q})$ is called from $\texttt{estimateSum}$, $\texttt{getDocuments}_V(\mathrm{Q})$ was already called in the same iteration by $\texttt{samplePair}(V)$. Thus, since documents are cached, the fetch cost of calling $\texttt{getDocuments}_V(\mathrm{Q})$ from $\texttt{estimateSum}$ is 0.

$\texttt{estimateInverseDegree}_V(\mathrm{X})$ fetches a single document (X). However, since this document was necessarily already fetched by $\texttt{getDocuments}_V(\mathrm{Q})$, and since documents are cached, the fetch cost of $\texttt{estimateInverseDegree}_V(\mathrm{X})$ is 0.

Obviously $\texttt{computeFunction}(\mathrm{X})$ performs a single function calculation. Thus the function cost of the estimator is n. □

8 Importance sampling estimator for averages

Recall that the estimator for averages estimates $\mathrm{avg}(f) = \mathrm{sum}_{\pi^n_{\mathcal{D}}}(f)$. Thus, in order to employ the importance sampling estimator (see Section 4.5), we need to compute the importance weight function

$$w_{\mathrm{avg}}(q, \mathrm{x}) \;=\; \frac{\pi^n_{\mathcal{D}}(\mathrm{x}) \cdot |\mathcal{P}_V| \cdot \deg_V(q)}{\deg_V(\mathrm{x})},$$

Notation	Meaning
AvgEst	The estimator for averages.
w_{avg}	The importance weight function of the estimator for averages.
WSE	The weight skew estimator.
ARIS	The approximate ratio importance sampling estimator.
(R, Y)	A random query-document pair distributed according to π^n.

Table 7: Notation used in Section 8.

where (q, x) is a sample from the trial distribution p.

However, since `computeTargetMeasure`(x) computes $\pi_{\mathcal{D}}(\mathrm{x})$, which equals $\pi^n_{\mathcal{D}}(\mathrm{x})$ up to normalization, we cannot calculate $\pi^n_{\mathcal{D}}(\mathrm{x})$ exactly. We overcome this difficulty by first estimating $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$, and then fixing the bias incurred by using $\pi_{\mathcal{D}}$ instead of $\pi^n_{\mathcal{D}}$.

Similarly to SumEst, we use the valid queries graph, and the same estimation procedures to compute the same importance weight function:

$$u(q, \mathrm{x}) \ = \ \pi_{\mathcal{D}}(\mathrm{x}) \cdot \mathrm{PSE} \cdot \deg_V(q) \cdot \mathrm{IDE}(\mathrm{x}).$$

(Recall that PSE is the estimator for $|\mathcal{P}_V|$ and $\mathrm{IDE}(\mathrm{x})$ is the estimator for $1/\deg_V(\mathrm{x})$.) Note that u is not an unbiased estimator for w_{avg}, since

$$\mathbb{E}(u(q, \mathrm{x})) \ = \ \mathbb{E}(Z_{\pi_{\mathcal{D}}} \cdot \pi^n_{\mathcal{D}}(\mathrm{x}) \cdot \mathrm{PSE} \cdot \deg_V(q) \cdot \mathrm{IDE}(\mathrm{x})) \ = \ Z_{\pi_{\mathcal{D}}} \cdot w_{\text{avg}}(q, \mathrm{x}), \tag{1}$$

where $Z_{\pi_{\mathcal{D}}}$ is the unknown normalization constant of $\pi_{\mathcal{D}}$.

By Theorem 4.4, using approximate importance weight function u instead of the exact one w_{avg} incurs a multiplicative bias $Z_{\pi_{\mathcal{D}}}$:

$$\mathbb{E}(\mathrm{AIS}(\mathrm{Q}, \mathrm{X})) \ = \ \mathrm{sum}_{\pi_{\mathcal{D}}}(f) \cdot Z_{\pi_{\mathcal{D}}} + Z_{\pi_{\mathcal{D}}} \cdot \mathrm{cov}(f(\mathrm{Y}), Z_{\pi_{\mathcal{D}}}) \ = \ \mathrm{sum}_{\pi_{\mathcal{D}}}(f) \cdot Z_{\pi_{\mathcal{D}}}.$$

In order to obtain an unbiased estimator for $\mathrm{sum}_{\pi_{\mathcal{D}}}(f)$ we still have to eliminate the $Z_{\pi_{\mathcal{D}}}$ factor. In the next subsection we propose a method for decreasing the bias.

8.1 Approximate ratio importance sampling

For an instance $\mathrm{x} \in \Omega$, the ratio $u(\mathrm{x})/w(\mathrm{x})$ is called the *weight skew at x*. The multiplicative bias factor is the expected weight skew relative to the target distribution π^n. In order to eliminate this bias, we need to somehow estimate the expected weight skew. For now, let us assume we have some unbiased weight skew estimator WSE(X) for $\mathbb{E}\left(\frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)$. (For example, in our case, $\mathbb{E}\left(\frac{u(\mathrm{R},\mathrm{Y})}{w_{\text{avg}}(\mathrm{R},\mathrm{Y})}\right) = Z_{\pi_{\mathcal{D}}}$, so $\mathrm{WSE}(\mathrm{Q}, \mathrm{X})$ should be an estimator for $Z_{\pi_{\mathcal{D}}}$.) Note that WSE(X) depends on the same sample X used by the importance sampling estimator. It follows from Theorem 4.4 that:

$$\frac{\mathbb{E}(\mathrm{AIS}(\mathrm{X}))}{\mathbb{E}(\mathrm{WSE}(\mathrm{X}))} \ = \ \mathrm{sum}_{\pi}(f) + \frac{Z_{\pi} \cdot \mathrm{cov}\left(f(\mathrm{Y}), \frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)}{\mathbb{E}\left(\frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)}.$$

Thus, the ratio of the expectations of the two estimators, AIS(X) and WSE(X), gives us the desired result ($\mathrm{sum}_\pi(f)$), modulo an additive bias factor. Ignoring for the moment this additive bias, it would seem that a good estimator for $\mathrm{sum}_\pi(f)$ is the ratio $\frac{\mathrm{AIS(X)}}{\mathrm{WSE(X)}}$. However, there is one problem: the expectation of a ratio is not the ratio of the expectations, i.e., $\mathbb{E}\left(\frac{\mathrm{AIS(X)}}{\mathrm{WSE(X)}}\right) \neq \frac{\mathbb{E}(\mathrm{AIS(X)})}{\mathbb{E}(\mathrm{WSE(X)})}$.

To solve this problem, we resort to a well-known trick from statistics: if we replace the numerator and the denominator by *averages* of multiple independent instances of the numerator estimator and of the denominator estimator, the difference between the expected ratio and the ratio of expectations diminishes to 0.

We can therefore define the approximate ratio importance sampling estimator for $\mathrm{sum}_\pi(f)$ as follows:

$$\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n) \;\triangleq\; \frac{\frac{1}{n}\sum_{i=1}^{n} \mathrm{AIS}(\mathrm{X}_i)}{\frac{1}{n}\sum_{i=1}^{n} \mathrm{WSE}(\mathrm{X}_i)},$$

where $\mathrm{X}_1, \ldots, \mathrm{X}_n$ are n independent samples from the trial distribution p. The following two lemmas analyze the bias and the variance of the estimator.

Lemma 8.1. *If* $\mathbb{E}(\mathrm{WSE}(X)) = \mathbb{E}\left(\frac{u(Y)}{w(Y)}\right)$, *then*

$$\mathbb{E}(\mathrm{ARIS}(X_1, \ldots, X_n)) - \mathrm{sum}_\pi(f) \;=\; \frac{Z_\pi \cdot \mathrm{cov}\left(f(Y), \frac{u(Y)}{w(Y)}\right)}{\mathbb{E}\left(\frac{u(Y)}{w(Y)}\right)} + O\left(\frac{1}{n}\right).$$

In Appendix B we provide the proof of the lemma and specify the high order term in the $O(1/n)$ expression. This term depends on $\mathrm{sum}_\pi(f)$, variance of u/w, and covariance of f and u/w. We conclude from the lemma that if we use sufficiently many samples, then we are likely to get an estimate of $\mathrm{sum}_\pi(f)$, which has only additive bias that depends on the correlation between f and u/w.

The next lemma shows that the variance of the estimator decreases to 0 as $1/n$.

Lemma 8.2.

$$\mathrm{var}(\mathrm{ARIS}(X_1, \ldots, X_n)) \;=\; O(1/n).$$

In Appendix B we provide the proof of the lemma and specify the high order term in the $O(1/n)$ expression. This term depends on $\mathrm{sum}_\pi(f)$, the bias of ARIS, the coefficients of variation of $f \cdot u$ and u/w, and the coefficient of covariation of $f \cdot u$ and u/w.

8.2 The estimator for averages

Recall (Section 4.3) that our estimators sample query-document pairs from the set of edges of the valid queries graph $\mathcal{E}_V$ and that π is the extension of $\pi_\mathcal{D}$ onto $\mathcal{E}_V$. To employ the approximate ratio importance sampling estimator described above, we need to come up with the weight skew estimator whose expectation equals $\mathbb{E}(u(\mathrm{R},\mathrm{Y})/w_{\mathrm{avg}}(\mathrm{R},\mathrm{Y}))$, where (R,Y) is distributed according to π. Observe that the approximate weight $u(\mathrm{Q},\mathrm{X})$ itself is an unbiased estimator for $Z_{\pi_\mathcal{D}}$, according to Theorem 4.4 with $f \equiv 1$ (the covariance term is 0 since f is a constant function). Since $\mathbb{E}(u(\mathrm{R},\mathrm{Y})/w_{\mathrm{avg}}(\mathrm{R},\mathrm{Y})) = Z_{\pi_{\mathcal{D}}}$, we set $\mathrm{WSE}(\mathrm{Q},\mathrm{X}) = u(\mathrm{Q},\mathrm{X})$.

AvgEst is then defined as follows:

$$\mathrm{AvgEst} = \frac{\frac{1}{n}\sum_{i=1}^{n} f(\mathrm{X}_i)\cdot u(\mathrm{Q}_i,\mathrm{X}_i)}{\frac{1}{n}\sum_{i=1}^{n} u(\mathrm{Q}_i,\mathrm{X}_i)},$$

where $(\mathrm{Q}_1,\mathrm{X}_1),\ldots,(\mathrm{Q}_n,\mathrm{X}_n)$ are n independent samples from the trial distribution p. Function 12 is the implementation of AvgEst. Note that the PSE factor in u cancels out since it appears in both the numerator and the denominator. Like in `estimateSum`, we cache all the query and document requests made by the estimator.

Function 12 `estimateAverage`

$\mathrm{sum}_{\pi_{\mathcal{D}}} := 0$
$\mathrm{WSE} := 0$
for $i := 1$ to n **do**
  $(\mathrm{Q},\mathrm{X}) :=$ `samplePair`(V)
  $\pi_{\mathcal{D}}(\mathrm{X}) :=$ `computeTargetMeasure`(X)
  $\deg_V(\mathrm{Q}) := |$`getDocuments`$_V(\mathrm{Q})|$
  $\mathrm{IDE}(\mathrm{X}) :=$ `estimateInverseDegree`$_V(\mathrm{X})$
  $u(\mathrm{Q},\mathrm{X}) := \pi_{\mathcal{D}}(\mathrm{X})\cdot\deg_V(\mathrm{Q})\cdot\mathrm{IDE}(\mathrm{X})$
  $\mathrm{sum}_{\pi_{\mathcal{D}}} := \mathrm{sum}_{\pi_{\mathcal{D}}} +$`computeFunction`$(\mathrm{X})\cdot u(\mathrm{Q},\mathrm{X})$
  $\mathrm{WSE} := \mathrm{WSE} + u(\mathrm{Q},\mathrm{X})$
return $\mathrm{sum}_{\pi_{\mathcal{D}}}\,/\,\mathrm{WSE}$

8.3 Analysis

By Lemma 8.1, the bias of the estimator is at most

$$\frac{Z_{\pi_{\mathcal{D}}}\cdot\mathrm{cov}\left(f(\mathrm{Y}),\frac{u(\mathrm{R},\mathrm{Y})}{w_{\mathrm{avg}}(\mathrm{R},\mathrm{Y})}\right)}{\mathbb{E}\left(\frac{u(\mathrm{R},\mathrm{Y})}{w_{\mathrm{avg}}(\mathrm{R},\mathrm{Y})}\right)} + O\left(\frac{1}{n}\right),$$

where $(\mathrm{R},\mathrm{Y}) \propto \pi$. Since $\mathbb{E}(u(\beta,\mathrm{y}))/w_{\mathrm{avg}}(\beta,\mathrm{y}) = Z_{\pi_{\mathcal{D}}}$ (see Equation 1), which is a constant, then $\mathrm{cov}\left(f(\mathrm{Y}),\frac{u(\mathrm{R},\mathrm{Y})}{w_{\mathrm{avg}}(\mathrm{R},\mathrm{Y})}\right) = 0$. Hence, we get:

Proposition 8.3. *The bias of AvgEst is $O(1/n)$.*

Proposition 8.4. *The costs of `estimateAverage` are equal to those of `estimateSum` (see Theorem 7.4).*

Proof. `estimateAverage` differs from `estimateSum` only by the calculation of WSE, which does not incur any additional costs. □

Notation	Meaning
EffSumEst	The efficient estimator for sums.
EffAvgEst	The efficient estimator for averages.
u_{eff}	The approximate importance weight function of the efficient estimators for sums and averages.

Table 8: Notation used in Section 9.

9 More efficient implementation of the estimators

In the estimators proposed in the previous sections we used accurate but rather expensive procedures for calculating importance weights. One of the main bottlenecks was the estimation of $1/\deg_V(\mathrm{x})$, which required submitting several queries to the search engine per each sample from the trial distribution. In the estimators we consider in this section, instead of using IDE(x) as an estimate of $1/\deg_V(\mathrm{x})$, we use $1/\deg_P(\mathrm{x})$. To compute $\deg_P(\mathrm{x})$ we need a single document fetch and no search engine queries. Unfortunately, since $\deg_P(\mathrm{x}) \neq \deg_V(\mathrm{x})$, the resulting estimator is biased. We analyze this bias and propose bias reduction techniques.

Recall that the importance weight function for SumEst is

$$w_{\text{sum}}(q, \mathrm{x}) \;=\; \frac{\pi_{\mathcal{D}}(\mathrm{x}) \cdot |\mathcal{P}_V| \cdot \deg_V(q)}{\deg_V(\mathrm{x})}$$

and the importance weight function for AvgEst is

$$w_{\text{avg}}(q, \mathrm{x}) \;=\; \frac{\pi^n_{\mathcal{D}}(\mathrm{x}) \cdot |\mathcal{P}_V| \cdot \deg_V(q)}{\deg_V(\mathrm{x})}.$$

We use the same approximate importance weight function u_{eff} for both the efficient estimator for sums and the efficient estimator for averages:

$$u_{\text{eff}}(q, \mathrm{x}) \;=\; \frac{\pi_{\mathcal{D}}(\mathrm{x}) \cdot \mathrm{PSE} \cdot \deg_V(q)}{\deg_P(\mathrm{x})}.$$

(Recall that PSE is the estimator for $|\mathcal{P}_V|$.)

We start with describing the efficient estimator for averages, and then derive from it the efficient estimator for sums.

Since u_{eff} is only an approximation of w_{avg}, we resort to approximate importance sampling (see Section 4.6). We now have to come up with a weight skew estimator. By Theorem 4.4 with $f \equiv 1$,

$$\mathbb{E}(u_{\text{eff}}(\mathrm{Q},\mathrm{X})) = \mathrm{sum}_{\pi^n_{\mathcal{D}}}(1) \cdot \mathbb{E}\left(\frac{u_{\text{eff}}(\mathrm{R},\mathrm{Y})}{w_{\text{avg}}(\mathrm{R},\mathrm{Y})}\right) + Z_{\pi^n_{\mathcal{D}}} \cdot \mathrm{cov}\left(1, \frac{u_{\text{eff}}(\mathrm{R},\mathrm{Y})}{w_{\text{avg}}(\mathrm{R},\mathrm{Y})}\right),$$

where $(\mathrm{Q},\mathrm{X}) \propto p$, $(\mathrm{R},\mathrm{Y}) \propto \pi^n$. Note that since $\pi^n_{\mathcal{D}}$ is a distribution, $\mathrm{sum}_{\pi^n_{\mathcal{D}}}(1) = Z_{\pi^n_{\mathcal{D}}} = 1$, and that the covariance term is 0 since 1 is a constant function. Thus,

$$\mathbb{E}(u_{\text{eff}}(\mathrm{Q},\mathrm{X})) = \mathbb{E}\left(\frac{u_{\text{eff}}(\mathrm{R},\mathrm{Y})}{w_{\text{avg}}(\mathrm{R},\mathrm{Y})}\right), \qquad (2)$$

and $u_{\text{eff}}(\mathrm{Q}, \mathrm{X})$ is the unbiased weight skew estimator we need.

EffAvgEst is then defined as follows:

$$\text{EffAvgEst} \;=\; \frac{\frac{1}{n}\sum_{i=1}^{n} f(\mathrm{X}_i) \cdot u_{\text{eff}}(\mathrm{Q}_i, \mathrm{X}_i)}{\frac{1}{n}\sum_{i=1}^{n} u_{\text{eff}}(\mathrm{Q}_i, \mathrm{X}_i)},$$

where $(\mathrm{Q}_1, \mathrm{X}_1), \ldots, (\mathrm{Q}_n, \mathrm{X}_n)$ are n independent samples from the trial distribution p.

Proposition 9.1. *The bias of EffAvgEst is:*

$$\frac{\operatorname{cov}\left(f(Y), \text{vdensity}(Y)\right)}{\mathbb{E}\left(\text{vdensity}(Y)\right)} + O\left(\frac{1}{n}\right),$$

where $Y \propto \pi_{\mathcal{D}}$.

Proof. The weight skew of the estimator is

$$\frac{u_{\text{eff}}(q, \mathrm{x})}{w_{\text{avg}}(q, \mathrm{x})} \;=\; Z_{\pi_{\mathcal{D}}} \cdot \frac{\deg_V(\mathrm{x})}{\deg_P(\mathrm{x})} \;=\; Z_{\pi_{\mathcal{D}}} \cdot \text{vdensity}(\mathrm{x}).$$

Substituting into Lemma 8.1 we get that the bias of the estimator is:

$$\frac{Z_{\pi_{\mathcal{D}}^n} \cdot \operatorname{cov}\left(f(\mathrm{Y}), Z_{\pi_{\mathcal{D}}} \cdot \text{vdensity}(\mathrm{Y})\right)}{\mathbb{E}\left(Z_{\pi_{\mathcal{D}}} \cdot \text{vdensity}(\mathrm{Y})\right)} + O\left(\frac{1}{n}\right),$$

Since $\pi_{\mathcal{D}}^n$ is a distribution, $Z_{\pi_{\mathcal{D}}^n} = 1$. $Z_{\pi_{\mathcal{D}}}$ cancels out and the lemma follows. □

We conclude that as long as the target function is not correlated with validity density, the bias is low.

9.1 Rao-Blackwellization

There is some inherent inefficiency in the importance sampling estimators: although each random query they submit to the search engine returns many results, they use at most a single result per query. All other results are discarded. The corpus size estimator of Broder *et al.* [8] uses all query results, and not just one. We observe that what they did is an instance of the well-known Rao-Blackwellization technique for reducing estimation variance. Thanks to the cheap importance weight computation we propose in this section, we can apply Rao-Blackwellization on our importance sampling estimators.

Recall that EffAvgEst repeatedly computes the basic approximate importance sampling estimator $\text{AIS}(\mathrm{Q}, \mathrm{X}) = f(\mathrm{X}) \cdot u_{\text{eff}}(\mathrm{Q}, \mathrm{X})$, where (Q, X) is a sample from the trial distribution p and $u_{\text{eff}}(\mathrm{Q}, \mathrm{X})$ is its approximate importance weight. In order to obtain the sample (Q, X), we choose a random query Q from $\mathcal{P}_V$ and then pick a single random document X from $\mathsf{documents}_V(\mathrm{Q})$, discarding other results. Suppose now that instead of using only this single document in the approximate importance sampling estimator, we use all the query results:

$$\text{AIS}^{\text{RB}}(\mathrm{Q}) \;=\; \frac{1}{\deg_V(\mathrm{Q})} \sum_{\mathrm{x} \in \mathsf{documents}_V(\mathrm{Q})} f(\mathrm{x}) \cdot u_{\text{eff}}(\mathrm{Q}, \mathrm{x}).$$

Each instance of AIS^{RB} is an average over several correlated instances of AIS. Mathematically, $\text{AIS}^{\text{RB}}(\mathrm{Q})$ is the conditional expectation of $\text{AIS}(\mathrm{Q},\mathrm{X})$ given Q:

$$\text{AIS}^{\text{RB}}(\mathrm{Q}) \;=\; \mathbb{E}(\text{AIS}(\mathrm{Q},\mathrm{X}) \mid \mathrm{Q}).$$

The main point is that computing these correlated instances in bulk can be done with a single search engine query. The Rao-Blackwell theorem implies that $\text{AIS}^{\text{RB}}(\mathrm{Q})$ can be only better than $\text{AIS}(\mathrm{Q},\mathrm{X})$ as an estimator for $\text{sum}_{\pi_{\mathcal{D}}}(f)$:

Theorem 9.2. AIS^{RB} *has the same bias as* AIS*:*

$$\mathbb{E}(\text{AIS}^{\text{RB}}(Q)) \;=\; \mathbb{E}(\text{AIS}(Q,X)).$$

The variance of AIS^{RB} *can only be lower:*

$$\text{var}(\text{AIS}^{\text{RB}}(Q)) \;=\; \text{var}(\text{AIS}(Q,X)) - \mathbb{E}(\text{var}(\text{AIS}(Q,X)|Q)).$$

Proof. Follows immediately from the Rao-Blackwell theorem (cf. [9]). □

By the above theorem, the expected reduction in variance is $\mathbb{E}(\text{var}(f(\mathrm{X}) \cdot u_{\text{eff}}(\mathrm{Q},\mathrm{X}))|\mathrm{Q})$, where Q is a uniformly chosen query from $\mathcal{P}_V$ and X is a uniformly chosen document from $\mathsf{documents}_V(\mathrm{Q})$. That is, the more variable are the results of queries w.r.t. the target function f, the higher are the chances that Rao-Blackwellization will help. The worst-case scenario is that all results of each query are the same, in which case Rao-Blackwellization does not reduce basic estimation variance at all. In our empirical study, however, we show that in practice Rao-Blackwellization can make a dramatic effect. See Section 10.

The variance reduction achieved by Rao-Blackwellization can lead to lower costs, as fewer instances of the estimator are needed in order to obtain a desired accuracy guarantee. On the other hand, each instance of the estimator requires many more weight and function calculations (as many as the number of results of the sampled query), and if these are very costly (as is the case with the estimators proposed in the previous sections), then the increase in cost per instance may outweigh the reduction in the number of instances, eventually leading to higher amortized costs. We conclude that Rao-Blackwellization should be used judiciously.

In our case, we expect query cost to be the main bottleneck, so using Rao-Blackwellization is justified. We thus redefine EffAvgEst as follows:

$$\text{EffAvgEst} \;=\; \frac{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{\deg_V(\mathrm{Q}_i)} \sum_{\mathrm{x} \in \mathsf{documents}_V(\mathrm{Q}_i)} f(\mathrm{x}) \cdot u_{\text{eff}}(\mathrm{Q}_i,\mathrm{x})}{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{\deg_V(\mathrm{Q}_i)} \sum_{\mathrm{x} \in \mathsf{documents}_V(\mathrm{Q}_i)} u_{\text{eff}}(\mathrm{Q}_i,\mathrm{x})},$$

where $\mathrm{Q}_1,\ldots,\mathrm{Q}_n$ are n uniform independent samples from $\mathcal{P}_V$. Function 13 is the implementation of EffAvgEst. Note that the PSE and the $\deg_V(\mathrm{Q}_i)$ factors in u_{eff} cancel out. Like in `estimateSum` and `estimateAverage`, we cache all the query and document requests made by the estimator.

We now analyze the query cost, the fetch cost, and the function cost of the estimator.

Function 13 `estimateAverageEfficiently`

$\text{sum}_{\pi_{\mathcal{D}}} := 0$
WSE $:= 0$
for $i := 1$ to n **do**
  $(\mathrm{Q}, \mathrm{X}) :=$ `samplePair`(V)
  documents$_V(\mathrm{Q}) :=$ `getDocuments`$_V(\mathrm{Q})$
  for $\mathrm{x} \in$ documents$_V(\mathrm{Q})$ **do**
    $\pi_{\mathcal{D}}(\mathrm{x}) :=$ `computeTargetMeasure`(x)
    $\deg_P(\mathrm{x}) :=$ `getDegree`$_P(\mathrm{x})$
    $u_{\text{eff}}(\mathrm{Q}, \mathrm{x}) := \pi_{\mathcal{D}}(\mathrm{x}) / \deg_P(\mathrm{x})$
    $\text{sum}_{\pi_{\mathcal{D}}} := \text{sum}_{\pi_{\mathcal{D}}} +$`computeFunction`$(\mathrm{x}) \cdot u_{\text{eff}}(\mathrm{Q}, \mathrm{x})$
    WSE $:=$ WSE $+ u_{\text{eff}}(\mathrm{Q}, \mathrm{x})$
return $\text{sum}_{\pi_{\mathcal{D}}}$ / WSE

Theorem 9.3.

$$\begin{aligned}
\mathbb{E}(\text{qcost}(\textit{estimateAverageEfficiently})) &= n \cdot \frac{|\mathcal{P}|}{|\mathcal{P}_V|}. \\
\mathbb{E}(\text{fetchcost}(\textit{estimateAverageEfficiently})) &= n \cdot \frac{|\mathcal{P}|}{|\mathcal{P}_V|} \cdot \text{avg}_{q \in \mathcal{P}} \deg_S(q). \\
\mathbb{E}(\text{funccost}(\textit{estimateAverageEfficiently})) &= n \cdot \text{avg}_{q \in \mathcal{P}_V} \deg_V(q).
\end{aligned}$$

Proof. Search engine queries are submitted in the following functions of the estimator:

- `estimatePoolSize`(V)
- `samplePair`(V)
- `getDocuments`$_V(\mathrm{Q})$

The query costs of these procedures were analyzed in the proof of the Theorem 7.4 and they are 0, $\frac{|\mathcal{P}|}{|\mathcal{P}_V|}$, and 0 respectively.

Documents are fetched in the following three functions of the estimator:

- `samplePair`(V)
- `getDocuments`$_V(\mathrm{Q})$
- `getDegree`$_P(\mathrm{x})$

The fetch costs of the first two procedures was analyzed in the proof of Theorem 7.4 and they are $|\mathcal{P}|/|\mathcal{P}_V| \cdot \text{avg}_{q \in \mathcal{P}} \deg_S(q)$ and 0, respectively. `getDegree`$_P(\mathrm{x})$ fetches a single document x. However, since this document was necessarily already fetched by `getDocuments`$_V(\mathrm{Q})$, and since documents are cached, the effective fetch cost of `getDegree`$_P(\mathrm{x})$ is 0.

It is easy to see that `computeFunction`(x) is called exactly $\deg_V(\mathrm{Q})$ times per external iteration of `estimateAverageEfficiently`. Since the sampled queries are distributed uniformly in $\mathcal{P}_V$, the expected function cost of `estimateAverageEfficiently` is

$$\mathbb{E}(\mathrm{funccost}(\texttt{estimateAverageEfficiently})) \;=\; \frac{1}{|\mathcal{P}_V|}\sum_{q\in\mathcal{P}_V}\deg_V(q) \;=\; \mathrm{avg}_{q\in\mathcal{P}_V}\deg_V(q).$$

□

9.2 Adaptation to estimation of sums

Recall that the "efficient" importance weight function is defined as follows:

$$u_{\mathrm{eff}}(q,\mathrm{x}) \;=\; \frac{\pi_{\mathcal{D}}(\mathrm{x})\cdot|\mathcal{P}_V|\cdot\deg_V(q)}{\deg_P(\mathrm{x})}.$$

Similarly to the efficient estimator for averages, since u_{eff} is only an approximation of w_{sum}, we use approximate importance sampling. In order to do this, we have to come up with an unbiased weight skew estimator whose expectation equals $\mathbb{E}\left(\frac{u_{\mathrm{eff}}(\mathrm{R},\mathrm{Y})}{w_{\mathrm{sum}}(\mathrm{R},\mathrm{Y})}\right)$, where $(\mathrm{R},\mathrm{Y}) \propto \pi^n$.

Unfortunately, $u_{\mathrm{eff}}(\mathrm{Q},\mathrm{X})$ itself is not an unbiased weight skew estimator (as it was for the efficient estimator for averages), since by Theorem 4.4 with $f \equiv 1$ (the covariance term is 0 since f is a constant function),

$$\mathbb{E}(u_{\mathrm{eff}}(\mathrm{Q},\mathrm{X})) \;=\; \mathrm{sum}_{\pi_{\mathcal{D}}}(1)\cdot\mathbb{E}\left(\frac{u_{\mathrm{eff}}(\mathrm{R},\mathrm{Y})}{w_{\mathrm{sum}}(\mathrm{R},\mathrm{Y})}\right) \;=\; Z_{\pi_{\mathcal{D}}}\cdot\mathbb{E}\left(\frac{u_{\mathrm{eff}}(\mathrm{R},\mathrm{Y})}{w_{\mathrm{sum}}(\mathrm{R},\mathrm{Y})}\right),$$

where $(\mathrm{Q},\mathrm{X}) \propto p$ and $(\mathrm{R},\mathrm{Y}) \propto \pi^n$. Note that $\mathrm{sum}_{\pi_{\mathcal{D}}}(1) = Z_{\pi_{\mathcal{D}}}$ is unknown. Suppose for the moment $Z_{\pi_{\mathcal{D}}}$ *is* known and thus can be used in the estimator.

EffSumEst can then be defined as follows:

$$\mathrm{EffSumEst} \;=\; Z_{\pi_{\mathcal{D}}}\cdot\frac{\frac{1}{n}\sum_{i=1}^n f(\mathrm{X}_i)\cdot u_{\mathrm{eff}}(\mathrm{Q}_i,\mathrm{X}_i)}{\frac{1}{n}\sum_{i=1}^n u_{\mathrm{eff}}(\mathrm{Q}_i,\mathrm{X}_i)},$$

where $(\mathrm{Q}_1,\mathrm{X}_1),\ldots,(\mathrm{Q}_n,\mathrm{X}_n)$ are n independent samples from the trial distribution p. Observe that EffSumEst is equal to EffAvgEst up to the factor $Z_{\pi_{\mathcal{D}}}$. Therefore, we can reduce EffSumEst to EffAvgEst as follows:

$$\mathrm{EffSumEst} \;=\; Z_{\pi_{\mathcal{D}}}\cdot\mathrm{EffAvgEst}\,.$$

$Z_{\pi_{\mathcal{D}}}$ can be estimated in a preprocessing step, e.g., by using SumEst with $f \equiv 1$. The cost of this one-time preprocessing can be amortized over multiple estimations that use the same target measure.

Proposition 9.4. *The bias of EffSumEst is*

$$\frac{Z_{\pi_{\mathcal{D}}}\cdot\mathrm{cov}\,(f(Y),\mathrm{vdensity}(Y))}{\mathbb{E}\,(\mathrm{vdensity}(Y))} + O\left(\frac{1}{n}\right),$$

where $Y \propto \pi^n_{\mathcal{D}}$.

Proof. The weight skew of the estimator is

$$\frac{u_{\text{eff}}(q, \mathrm{x})}{w_{\text{sum}}(q, \mathrm{x})} = \frac{\deg_V(\mathrm{x})}{\deg_P(\mathrm{x})} = \text{vdensity}(\mathrm{x}).$$

The lemma follows by substituting this into Lemma 8.1. □

The costs of the estimator are equivalent to those of `estimateAverageEfficiently` (see Theorem 9.3) up to the costs of estimating $Z_{\pi_{\mathcal{D}}}$. Indeed, if an estimate of $Z_{\pi_{\mathcal{D}}}$ is not available in advance, `estimateSumEfficiently` is not more efficient than `estimateSum`. However, in many practical situations, estimate of $Z_{\pi_{\mathcal{D}}}$ is available as a byproduct of another estimation, so that its cost for `estimateAverageEfficiently` is zero.

10 Experimental results

We conducted two sets of experiments. In the first set we performed comparative evaluation of the bias and amortized cost of our new estimators, of the rejection sampling estimator from our previous paper [5], and of the Broder *et al.* estimator [8]. To this end, we ran all these estimators on a local search engine that we built over 2.4 million English documents fetched from ODP [12]. As we have ground truth for this search engine, we could compare the measurements produced by the estimators against the real values.

The second set of experiments was conducted over two major real search engines. We used SumEst to estimate the corpus size of each the search engines, with and without duplicate elimination. (More accurately, we estimated sizes of large subsets of the search engine corpora.). We then used AvgEst to estimate the index freshness, the fraction of pages containing advertisements, and the distribution of web server types.

10.1 Evaluation experiments

Experimental setup. We used the same local search engine as in our previous paper [5]. The corpus of this search engine consists of English-language text, HTML, and pdf documents from the ODP hierarchy. Each document was given a serial id and indexed by single terms and phrases. Only the first 10,000 terms in each document were considered. Exact phrases were not allowed to cross boundaries, such as paragraph boundaries. We used static ranking by serial id to rank query results.

In order to construct a query pool for the evaluation experiments, we split the ODP data set into two parts: a *training set*, consisting of every fifth page (when ordered by id), and a *test set*, consisting of the rest of the pages. We used the training set to create a pool of phrases of length 4. The measurements were done only on the test set.

We compared the following 6 estimator configurations: (1) SumEst; (2) EffSumEst; (3) AvgEst; (4) EffAvgEst; (5) the *Broder et al.* estimator; (6) the rejection sampling estimator from our previous paper.

We used the estimators to measure two metrics: (1) corpus size (i.e., the size of the test set); (2) density of pages in the test set about sports (we used a simple keyword based classifier to determine whether a page is about sports or not). Note that corpus size is a sum metric, while the density of sports pages is an average metric. We did not use the rejection sampling estimator for estimating corpus size, as it can handle only average metrics. We did not use the Broder *et al.* estimator for estimating the density of sports pages, because it can handle only sum metrics.

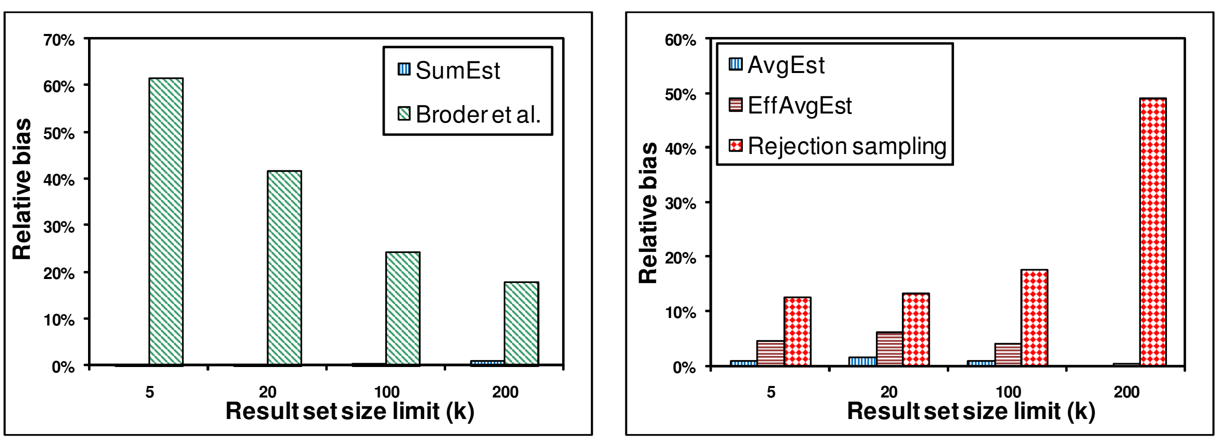
In order to have a common baseline, we allowed each estimator to sample exactly 1 million non-underflowing queries from the pool. Each estimator then submitted all these queries to the local search engine and computed its estimate. Some estimator used additional search engine requests for estimating document degrees.

We ran each experiment four times, with different values of the result set size limit k ($k = 5, 20, 100, 200$). This was done in order to track the dependence of the estimator bias on the validity density (the lower is k, the higher this density is expected to be), and the dependence of the estimator cost on the result set sizes (which increase with k).

For each produced estimate, we measured relative bias and amortized cost as follows. Let E be the estimation result and let I be the true value of the parameter being estimated. The relative bias is $|E - I|/I$.

The amortized query cost of an estimator M is $\mathrm{qcost}(\mathrm{M}) \cdot \frac{\mathrm{var}(\mathrm{M})}{E^2(\mathrm{M})}$ (see Section 3.2). We used the total number of queries made by the estimator during its execution as an estimate of qcost(M). The measured empirical variance of M was used as an estimate of var(M), and M's output was used as an estimate of E(M).

Results.



(a) Corpus size.

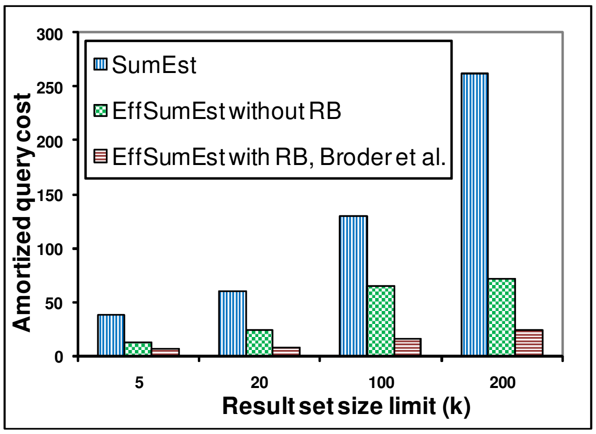
(b) Density of sports pages.

Figure 1: Relative bias of the estimators.

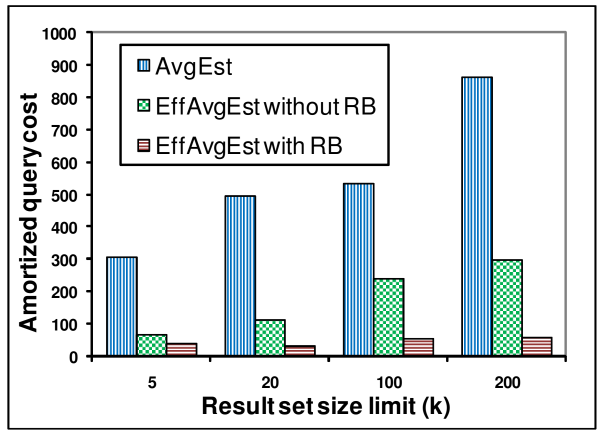
Figure 1(a) compares the relative bias of SumEst and the estimator of Broder *et al.* when measuring corpus size (for corpus size estimation, EffSumEst is equivalent to SumEst, and is therefore omitted from this figure). Figure 1(b) compares the relative bias of our two estimators and the rejection sampling estimator when measuring density of sports pages.

The results for the corpus size clearly show that our estimator has no bias at all, while the estimator of Broder *et al.* suffers from significant bias, which grows with the density of overflowing queries in the pool. For example, for $k = 5$, the relative bias of the Broder *et al.* estimator is about 60%, while the relative bias of our estimators below 1%.

The results for the density of sports pages show that AvgEst is practically unbiased, as expected. EffAvgEst has small bias, which emanates from a weak correlation between the function value and the validity density. The rejection sampling method has a large observed bias, primarily because it produced a small number of uniform samples and thus its variance is still high.



(a) Sum metric.



(b) Average metric.

Figure 2: Amortized query cost of the estimators.

Figures 2(a) and 2(b) compare the amortized query costs of our estimators, and demonstrate the effect of Rao-Blackwellization on the query cost of the efficient estimators. The cost of the estimator of Broder *et al.* is the same as the cost of EffSumEst with Rao-Blackwellization. The amortized query costs of the rejection sampling estimator were 5600, 14100, 80200, 84700 for $k = 5$, 20, 100, 200 respectively. They are omitted from Figure 2(b) as they are significantly higher than the costs of the other estimators and would distort the plot. The cost of EffSumEst does not include the cost of the preprocessing step.

The amortized query cost increases with k due to increasing variance of query degrees (see Section 6.2). The results clearly indicate that Rao-Blackwellization is effective in reducing estimation variance (and therefore also amortized cost) in all estimators. For example, in the estimation of the density of sports pages, when $k = 200$, Rao-Blackwellization reduced the amortized query cost of EffAvgEst by 80%. Furthermore, the amortized cost of the rejection sampling estimator is tremendously higher than the amortized cost of our new estimators (even the non-Rao-Blackwellized ones). For example, when $k = 100$, Rao-Blackwellized EffAvgEst was 1,500 times more efficient than rejection sampling!

10.2 Experiments on real search engines

Experimental setup. The experiments on real search engines were conducted in July 2009. Unlike our previous works [5, 3], where we used phrase queries, here we used *inurl* queries (i.e., `"inurl:<term>"`) of single terms and two-term conjunctions. Inurl queries return only pages whose *URL* contains the query terms. This is different from regular queries which return pages based on matching content, URL, anchor text, etc. The main advantage of the pool of inurl queries over the pool we have used before is that it covers more types of documents (images, flash, etc.) and non-English documents. Note that this pool too is not guaranteed to cover 100% of all the pages in the corpus. For example, pages whose URLs consist only of non-English terms are less likely to be covered.

The pool used by our estimators was a pool of 2.37 billion queries consisting of: (1) 1.11 billion decimal strings of 5 to 9 digits, (2) 7.26 million single terms extracted from snapshots of the English Wikipedia site and the English ODP directory, (3) 10.7 million single terms extracted from the list of URLs in the ODP directory, (4) 1.25 billion two-term conjunctions of the 50,000 most frequent single terms (excluding the 100 most frequent ones) from (2) and (3).

To further increase corpus coverage we disabled the standard result filtering performed by the engines (duplicate filtering and host collapsing) by adding suitable arguments to the requests we sent to the search engines.

Corpus size. We used our most accurate sampler, SumEst, to estimate the corpus sizes of two major search engines. For reference, we also ran the Broder *et al.* estimator with the same query pool. In addition to the corpus measurement with result filtering disabled, we performed additional measurements with both duplicate filtering and host collapsing enabled, which is the default setting for regular web search. The results and their standard deviations are plotted in Figure 3.

We note that our estimates may underestimate the true corpus sizes of the search engines since they effectively measure the sizes of only subsets of the corpora—the indexed pages that match at least one query from the pool.

The results show that the degree mismatch problem affects the Broder *et al.* estimator also on live search engines. Its estimate of the corpus size of the first search engine was 3.8 times lower than our estimate, and its estimate of the corpus size of the second search engine was 3.5 times lower than our estimate.

Finally, the experiments reveal that duplicate filtering and host collapsing make about half of the corpora "invisible" for the queries in our pool.

Corpus freshness. We used AvgEst to estimate the percentage of dead pages (ones returning a 4xx HTTP return code) in the full corpora (with both duplicate filtering and host collapsing disabled). The results and their standard deviations are plotted in Figure 4. Observe that both corpora have a comparable, yet non-negligible, fraction of inaccessible pages.

Ads. We used AvgEst to estimate the fraction of the indexed pages containing at least one ad. We used a simple ad detection heuristic that checked whether the page contains a string of the form `"http://..."` (including quotes) containing one of the following substrings: `googlesyndication,`

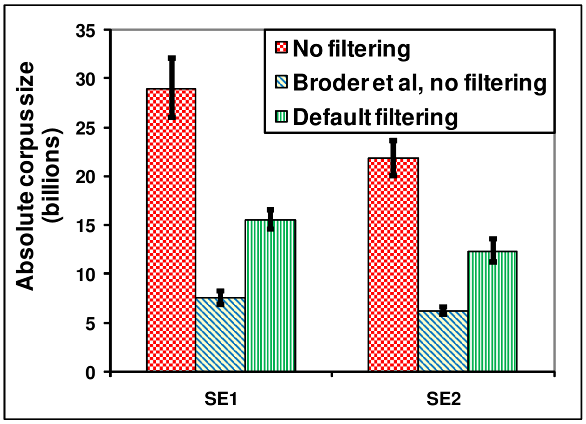


Figure 3: Corpus sizes of two major search engines, with and without host collapsing and elimination of duplicates.

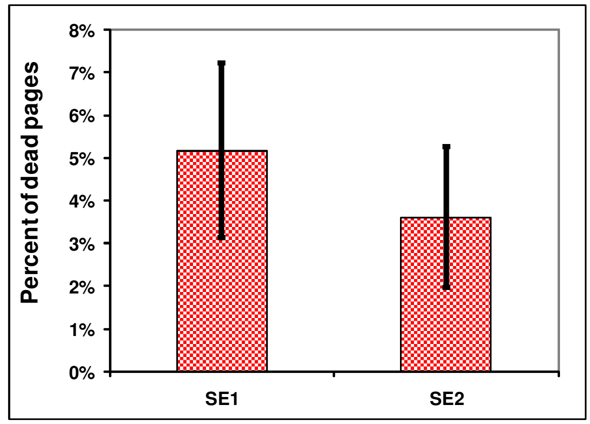


Figure 4: Percentage of inaccessible pages.

googlesyndication, googleadservices, doubleclick, adsense, omniture, atdmt, aolcdn, eiv.baidu.com, ma.baidu.com, /spcjs.php?, /ck.php?, adserver.yahoo.com, adsfac.net, ad.yieldmanager.com, adbureau.net, ads.revsci.net, blogads, or one of the regular expressions /ads\W, /ad\W. The results and their standard deviations are plotted in Figure 5. The results indicate that more than 30% of the indexed pages contains at least one ad.

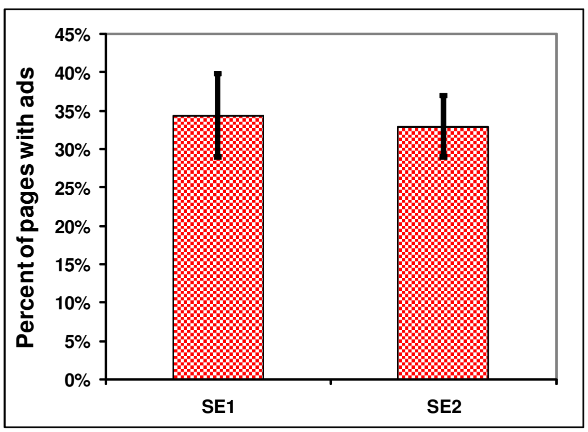


Figure 5: Percentage of pages containing ads.

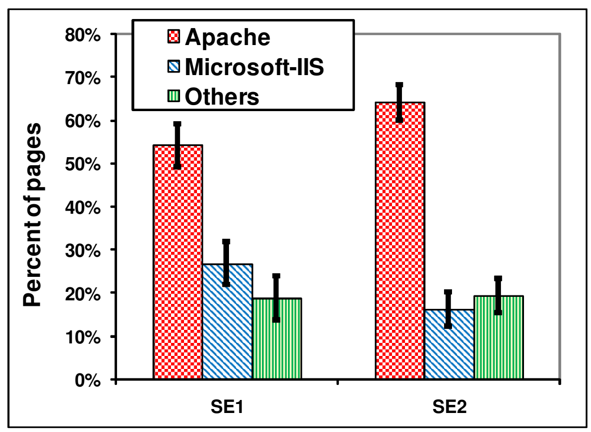


Figure 6: Distribution of web server types hosting the indexed pages.

Web server types. Finally, we used AvgEst to estimate the distribution of web server types hosting the indexed pages. We used the "Server" attribute of the http response to determine the server type. The results and their standard deviations are plotted in Figure 6. Apache-based servers clearly host the majority of the indexed pages while Microsoft-IIS is a distant second. Other server types are below the error margin of our measurements and are thus aggregated under category "Others".

11 Conclusions

In this paper we presented two new estimators for aggregate function over search engine corpora that can be expressed as discrete integrals. Our estimators are able to overcome the "degree mismatch" problem and thereby be accurate and efficient at the same time. We show both analytically and empirically that our estimators beat recently proposed estimators [5, 8].

In designing our estimators we employ a combination of statistical tools, like importance sampling and Rao-Blackwellization. By carefully analyzing the effect of approximate weights on the bias of importance sampling, we were able to design procedures to mitigate the bias. This bias-elimination technique for approximate importance sampling may be applicable in other scenarios as well.

References

[1] A. Anagnostopoulos, A. Broder, and D. Carmel. Sampling search-engine results. In *Proc. 14th WWW*, pages 245–256, 2005.

[2] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about Web pages via random walks. In *Proc. 26th VLDB*, pages 535–544, 2000.

[3] Z. Bar-Yossef and M. Gurevich. Efficient search engine measurements. In *Proceedings of the 16th International World Wide Web Conference (WWW)*, pages 401–410, 2007.

[4] Z. Bar-Yossef and M. Gurevich. Mining search engine query logs via suggestion sampling. In *Proc. 34th VLDB*, 2008.

[5] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. *J. ACM*, 55(5), 2008.

[6] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public Web search engines. In *Proc. 7th WWW*, pages 379–388, 1998.

[7] E. T. Bradlow and D. C. Schmittlein. The little engines that could: Modeling the performance of World Wide Web search engines. *Marketing Science*, 19:43–62, 2000.

[8] A. Broder, M. Fontoura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu. Estimating corpus size via queries. In *Proc. 15th CIKM*, 2006.

[9] G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[10] M. Cheney and M. Perry. A comparison of the size of the Yahoo! and Google indices. Available at `http://vburton.ncsa.uiuc.edu/indexsize.html`, 2005.

[11] A. Dasgupta, G. Das, and H. Mannila. A random walk approach to sampling hidden databases. In *Proc. SIGMOD*, pages 629–640, 2007.

[12] dmoz. The open directory project. `http://dmoz.org`.

[13] A. Dobra and S. E. Fienberg. How large is the World Wide Web? *Web Dynamics*, pages 23–44, 2004.

[14] O. Goldreich. A sample of samplers - a computational perspective on sampling (survey). *ECCC*, 4(20), 1997.

[15] A. Gulli and A. Signorini. The indexable Web is more than 11.5 billion pages. In *Proc. 14th WWW*, pages 902–903, 2005.

[16] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[17] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring index quality using random walks on the Web. In *Proc. 8th WWW*, pages 213–225, 1999.

[18] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. In *Proc. 9th WWW*, pages 295–308, 2000.

[19] T. C. Hesterberg. *Advances in Importance Sampling*. PhD thesis, Stanford University, 1988.

[20] S. Lawrence and C. L. Giles. Searching the World Wide Web. *Science*, 5360(280):98, 1998.

[21] S. Lawrence and C. L. Giles. Accessibility of information on the Web. *Nature*, 400:107–109, 1999.

[22] S.-M. Lee and A. Chao. Estimating population size via sample coverage for closed capture-recapture models. *Biometrics*, 50(1):88–97, 1994.

[23] J. S. Liu. Monte Carlo Strategies in Scientific Computing. Springer, 2001.

[24] A. W. Marshall. The use of multi-stage sampling schemes in Monte Carlo computations. In *Symposium on Monte Carlo Methods*, pages 123–140, 1956.

[25] K. McCurley. Income inequality in the attention economy. `http://mccurley.org/papers/effective`, 2007.

[26] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. of Chemical Physics*, 21:1087–1091, 1953.

[27] P. Rusmevichientong, D. Pennock, S. Lawrence, and C. L. Giles. Methods for sampling pages uniformly from the World Wide Web. In *Proc. AAAI Symp. on Using Uncertainty within Computation*, 2001.

[28] D. Siegmund. *Sequential Analysis - Tests and Confidence Intervals*. Springer-Verlag, 1985.

[29] A. Stuart and J. K. Ord. *Kendall's advanced theory of statistics. Vol.1: Distribution theory.* London: Hodder Arnold, 1994.

[30] J. von Neumann. Various techniques used in connection with random digits. In *John von Neumann, Collected Works*, volume V. Oxford, 1963.

A Search engine importance sampling estimation

Importance sampling with approximate degrees

Theorem 4.4 (restated)

$$\mathbb{E}(\mathrm{AIS}(X)) \;=\; \mathrm{sum}_\pi(f) \cdot \mathbb{E}\left(\frac{u(Y)}{w(Y)}\right) + Z_\pi \cdot \mathrm{cov}\left(F(Y), \frac{u(Y)}{w(Y)}\right),$$

where X is distributed according to the trial distribution p, Y is distributed according to the target distribution π^n, and Z_π is the normalization constant of π.

Proof. Since $\mathrm{supp}(u) \subseteq \mathrm{supp}(w)$,

$$\begin{aligned}
\mathbb{E}(\mathrm{AIS}(\mathrm{X})) &= \mathbb{E}(f(\mathrm{X}) \cdot u(\mathrm{X})) \\
&= \sum_{\mathrm{x} \in \mathrm{supp}(u)} p(\mathrm{x})\ f(\mathrm{x})\ u(\mathrm{x}) \\
&= \sum_{\mathrm{x} \in \mathrm{supp}(w)} p(\mathrm{x})\ f(\mathrm{x})\ w(\mathrm{x})\ \frac{u(\mathrm{x})}{w(\mathrm{x})} \\
&= \sum_{\mathrm{x} \in \mathrm{supp}(w)} p(\mathrm{x})\ f(\mathrm{x})\ \frac{\pi(\mathrm{x})}{p(\mathrm{x})}\ \frac{u(\mathrm{x})}{w(\mathrm{x})} \\
&= \sum_{\mathrm{y} \in \mathrm{supp}(w)} \pi(\mathrm{y})\ f(\mathrm{y})\ \frac{u(\mathrm{y})}{w(\mathrm{y})} \quad \text{(Renaming variables x} \to \text{y)} \\
&= \sum_{\mathrm{y} \in \mathrm{supp}(w)} Z_\pi\ \pi^n(\mathrm{y})\ f(\mathrm{y})\ \frac{u(\mathrm{y})}{w(\mathrm{y})} \\
&= Z_\pi\, \mathbb{E}\left(f(\mathrm{Y})\ \frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right) \\
&= Z_\pi \left(\mathbb{E}(f(\mathrm{Y})) \cdot \mathbb{E}\left(\frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right) + \mathrm{cov}\left(f(\mathrm{Y}), \frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)\right) \\
&= \mathrm{sum}_\pi(f(\mathrm{Y})) \cdot \mathbb{E}\left(\frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right) + Z_\pi \cdot \mathrm{cov}\left(f(\mathrm{Y}), \frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right).
\end{aligned}$$

□

B Importance sampling estimator for averages

Approximate importance sampling

Below we define the *ratio estimator* and show that its bias and variance decrease to 0 as $1/n$.

Definition B.1 (Ratio estimator). *Suppose M and N are two estimators such that $\frac{\mathbb{E}(M)}{\mathbb{E}(N)} = I$. Let $M_1, \ldots, M_n$ and $N_1, \ldots, N_n$ be n independent instances of M and N, respectively. Define $\overline{M}_n \triangleq \frac{1}{n}\sum_{i=1}^{n} M_i$ and $\overline{N}_n \triangleq \frac{1}{n}\sum_{i=1}^{n} N_i$. Then,*

$$\mathrm{RE}_n \;=\; \frac{\overline{M}_n}{\overline{N}_n}$$

is a ratio estimator *of order n induced by M and N.*

Theorem B.2. *Assume that M and N have finite variance and that $\overline{N}_n > 0$. Then, the bias of RE_n is:*

$$\mathbb{E}(\mathrm{RE}_n) - I \;=\; \frac{1}{n}\cdot\left(I\cdot\frac{\mathrm{var}(N)}{\mathbb{E}^2(N)} + \frac{\mathrm{cov}(M,N)}{\mathbb{E}^2(N)}\right) + o(1/n).$$

Proof. We use the Delta method in statistics (see, e.g., [29], section 10.5, pages 350,371) to approximate $\mathbb{E}(\mathrm{RE}_n)$.

$$\mathbb{E}(\mathrm{RE}_n) \;=\; \mathbb{E}\left(\frac{\overline{\mathrm{M}}_n}{\overline{\mathrm{N}}_n}\right) = \frac{\mathbb{E}(\overline{\mathrm{M}}_n)}{\mathbb{E}(\overline{\mathrm{N}}_n)} + \frac{\mathbb{E}(\overline{\mathrm{M}}_n)}{\mathbb{E}^3(\overline{\mathrm{N}}_n)}\,\mathrm{var}(\overline{\mathrm{N}}_n) - \frac{\mathrm{cov}(\overline{\mathrm{M}}_n,\overline{\mathrm{N}}_n)}{\mathbb{E}^2(\overline{\mathrm{N}}_n)} + o(1/n).$$

Since $\mathbb{E}(\overline{\mathrm{M}}_n) = \mathbb{E}(\mathrm{M})$, $\mathbb{E}(\overline{\mathrm{N}}_n) = \mathbb{E}(\mathrm{N})$, and $\mathrm{cov}(\overline{\mathrm{M}}_n,\overline{\mathrm{N}}_n) = \mathrm{cov}(\mathrm{M},\mathrm{N})/n$, we simplify it as follows:

$$\mathbb{E}\left(\frac{\overline{\mathrm{M}}_n}{\overline{\mathrm{N}}_n}\right) \;=\; I + \frac{1}{n}\cdot\left(I\cdot\frac{\mathrm{var}(\mathrm{N})}{\mathbb{E}^2(\mathrm{N})} - \frac{\mathrm{cov}(\mathrm{M},\mathrm{N})}{\mathbb{E}^2(\mathrm{N})}\right) + o(1/n).$$

□

Note that the bias decreases to 0 as $1/n$, and depends on the coefficient of variation of N (the standard deviation of N divided by its expectation) as well as on the correlation between M and N. If N has low variance and M and N have low correlation (e.g., if they are independent), then few instances will be needed in order to make the bias small.

The next theorem quantifies the variance of the ratio estimator.

Theorem B.3. *Assume that M and N have finite variance and that $\overline{N}_n > 0$. Then, the variance of RE_n is:*

$$\mathrm{var}(\mathrm{RE}_n) \;=\; I^2\cdot\frac{1}{n}\cdot\left(\frac{\mathrm{var}(M)}{\mathbb{E}^2(M)} + \frac{\mathrm{var}(N)}{\mathbb{E}^2(N)} - \frac{2\,\mathrm{cov}(M,N)}{\mathbb{E}(M)\,\mathbb{E}(N)}\right) + o(1/n).$$

Proof. Here too, we use the Delta method in statistics (see, e.g., [29], section 10.5, pages 350–351) to approximate $\mathrm{var}(\mathrm{RE}_n)$.

$$\mathrm{var}(\mathrm{RE}_n) \;=\; \left(\frac{\mathbb{E}(\overline{\mathrm{M}}_n)}{\mathbb{E}(\overline{\mathrm{N}}_n)}\right)^2\left(\frac{\mathrm{var}(\overline{\mathrm{M}}_n)}{\mathbb{E}^2(\overline{\mathrm{M}}_n)} + \frac{\mathrm{var}(\overline{\mathrm{N}}_n)}{\mathbb{E}^2(\overline{\mathrm{N}}_n)} - \frac{2\,\mathrm{cov}(\overline{\mathrm{M}}_n,\overline{\mathrm{N}}_n)}{\mathbb{E}(\overline{\mathrm{M}}_n)\,\mathbb{E}(\overline{\mathrm{N}}_n)}\right) + o(1/n),$$

which is further simplified as follows:

$$\mathrm{var}(\mathrm{RE}_n) = I^2\cdot\frac{1}{n}\cdot\left(\frac{\mathrm{var}(\mathrm{M})}{\mathbb{E}^2(\mathrm{M})} + \frac{\mathrm{var}(\mathrm{N})}{\mathbb{E}^2(\mathrm{N})} - \frac{2\,\mathrm{cov}(\mathrm{M},\mathrm{N})}{\mathbb{E}(\mathrm{M})\,\mathbb{E}(\mathrm{N})}\right) + o(1/n).$$

□

Similarly to the bias, the variance decreases to 0 as $1/n$. It can be seen that the variance depends on the coefficients of variation of M and N and on their coefficient of covariation.

Lemma 8.1 (restated) *If* $\mathbb{E}(\mathrm{WSE}(X)) = \mathbb{E}\left(\frac{u(Y)}{w(Y)}\right)$, *then*

$$\begin{aligned}&\mathbb{E}(\mathrm{ARIS}(X_1, \ldots, X_n)) - \mathrm{sum}_\pi(f) \\ &\quad = \; C \; + \; \frac{1}{n} \cdot \left((\mathrm{sum}_\pi(f) + C) \cdot \frac{\mathrm{var}(\mathrm{WSE}(X))}{\mathbb{E}^2(\mathrm{WSE}(X))} \; + \; \frac{\mathrm{cov}(\mathrm{AIS}(X), \mathrm{WSE}(X))}{\mathbb{E}^2(\mathrm{WSE}(X))} \right) + o(1/n),\end{aligned}$$

where

$$C \;=\; \frac{Z_\pi \cdot \mathrm{cov}\left(f(Y), \frac{u(Y)}{w(Y)}\right)}{\mathbb{E}\left(\frac{u(Y)}{w(Y)}\right)}.$$

Proof. From Theorems 4.4 and B.2,

$$\begin{aligned}&\mathbb{E}(\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n)) \\ &\quad = \; \mathrm{sum}_\pi(f) \; + \; \frac{Z_\pi \cdot \mathrm{cov}\left(f(\mathrm{Y}), \frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)}{\mathbb{E}\left(\frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)} \\ &\quad + \; \frac{1}{n} \cdot \left(\mathbb{E}(\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n)) \cdot \frac{\mathrm{var}(\mathrm{WSE}(\mathrm{X}))}{\mathbb{E}^2(\mathrm{WSE}(\mathrm{X}))} + \frac{\mathrm{cov}(\mathrm{AIS}(\mathrm{X}), \mathrm{WSE}(\mathrm{X}))}{\mathbb{E}^2(\mathrm{WSE}(\mathrm{X}))} \right) + o(1/n).\end{aligned}$$

The lemma follows by substituting

$$\mathbb{E}(\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n)) \;=\; \mathrm{sum}_\pi(f) \;+\; \frac{Z_\pi \cdot \mathrm{cov}\left(f(\mathrm{Y}), \frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)}{\mathbb{E}\left(\frac{u(\mathrm{Y})}{w(\mathrm{Y})}\right)} + O\left(\frac{1}{n}\right),$$

into the above expression. □

Lemma 8.2 (restated) *Let C be the highest order term of the bias of the* ARIS *estimator:*

$$C \;=\; \frac{Z_\pi \cdot \mathrm{cov}\left(f(Y), \frac{u(Y)}{w(Y)}\right)}{\mathbb{E}\left(\frac{u(Y)}{w(Y)}\right)}.$$

Then,

$$\begin{aligned}&\mathrm{var}(\mathrm{ARIS}(X_1, \ldots, X_n)) \\ &\quad = \; \frac{1}{n} \cdot (\mathrm{sum}_\pi(f) + C)^2 \\ &\qquad \cdot \left(\frac{\mathrm{var}(\mathrm{AIS}(X))}{\mathbb{E}^2(\mathrm{AIS}(X))} + \frac{\mathrm{var}(\mathrm{WSE}(X))}{\mathbb{E}^2(\mathrm{WSE}(X))} - \frac{2\,\mathrm{cov}(\mathrm{AIS}(X), \mathrm{WSE}(X))}{\mathbb{E}(\mathrm{AIS}(X)) \cdot \mathbb{E}(\mathrm{WSE}(X))} \right) + o(1/n).\end{aligned}$$

Proof. By Theorem B.3,

$$\begin{aligned}
&\mathrm{var}(\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n)) \\
&= \mathbb{E}^2(\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n)) \cdot \frac{1}{n} \cdot \left(\frac{\mathrm{var}(\mathrm{AIS}(\mathrm{X}))}{\mathbb{E}^2(\mathrm{AIS}(\mathrm{X}))} + \frac{\mathrm{var}(\mathrm{WSE}(\mathrm{X}))}{\mathbb{E}^2(\mathrm{WSE}(\mathrm{X}))} - \frac{2\,\mathrm{cov}(\mathrm{AIS}(\mathrm{X}), \mathrm{WSE}(\mathrm{X}))}{\mathbb{E}(\mathrm{AIS}(\mathrm{X})) \cdot \mathbb{E}(\mathrm{WSE}(\mathrm{X}))} \right) \\
&\quad + o(1/n).
\end{aligned}$$

We restate the first term as follows:

$$\begin{aligned}
\mathbb{E}^2(\mathrm{ARIS}(\mathrm{X}_1, \ldots, \mathrm{X}_n)) &= \left(\mathrm{sum}_\pi(f) + C + O\left(\frac{1}{n}\right) \right)^2 \\
&= (\mathrm{sum}_\pi(f) + C)^2 + O(1/n).
\end{aligned}$$

The lemma follows. $\square$