Ranking Deep Web Text Collections for Scalable Information Extraction

Pablo Barrio
Columbia University
New York, NY, USA
pjbarrio@cs.columbia.edu

Luis Gravano
Columbia University
New York, NY, USA
gravano@cs.columbia.edu

Chris Develder
Ghent University – iMinds
Ghent, Belgium
cdvelder@intec.ugent.be

ABSTRACT

Information extraction (IE) systems discover structured information from natural language text, to enable much richer querying and data mining than possible directly over the unstructured text. Unfortunately, IE is generally a computationally expensive process, and hence improving its efficiency, so that it scales over large volumes of text, is of critical importance. State-of-the-art approaches for scaling the IE process focus on one text collection at a time. These approaches prioritize the extraction effort by learning keyword queries to identify the "useful" documents for the IE task at hand, namely, those that lead to the extraction of structured "tuples." These approaches, however, do not attempt to predict which text collections are useful for the IE task—and hence merit further processing—and which ones will not contribute any useful output—and hence should be ignored altogether, for efficiency. In this paper, we focus on an especially valuable family of text sources, the so-called deep web collections, whose (remote) contents are only accessible via querying. Specifically, we introduce and study techniques for ranking deep web collections for an IE task, to prioritize the extraction effort by focusing on collections with substantial numbers of useful documents for the task. We study both (adaptations of) state-of-the-art resource selection strategies for distributed information retrieval, and IE-specific approaches. Our extensive experimental evaluation over realistic deep web collections, and for several different IE tasks, shows the merits and limitations of the alternative families of approaches, and provides a roadmap for addressing this critically important building block for efficient, scalable information extraction.

1. INTRODUCTION

A large body of today's knowledge is embedded in natural language text documents. Information extraction (IE) systems identify and extract information from text into a structured form, such as relational tables. For example, an IE system to extract an *Occurs-in*(*Natural Disaster*, *Location*) relation would extract the tuple ⟨earthquake, Mexico⟩ from the text fragment *"A powerful earthquake shook a wide area of Mexico."* Much richer querying and data mining are possible over structured information than over the unstructured text where it originates. Unfortunately, IE is generally a computationally expensive process [22], and hence improving its efficiency, so that it scales to large volumes of text, is of critical importance.

In this paper, we focus on IE over *deep web collections*, which are text document collections that are accessible only through a query interface and are hence not "crawlable" through traditional search engine crawlers [7, 17, 27, 28]. Examples of such collections include the Federal Emergency Management Agency (FEMA) collection[1], a key up-to-date resource for natural disasters and other hazards in the United States; and PubMed[2], a well-known resource for life sciences and biomedical research with over 22 million abstracts and references to research papers. Deep web collections often host high-quality content and together span a broad range of topics. Furthermore, the collective content in deep web collections may exceed in volume, according to some estimates, that of the crawlable, or "surface" web [18, 24].

To run an IE system over a deep web collection, a key challenge is effectively and efficiently retrieving its *useful* documents, namely, the documents from which the IE system manages to extract tuples. To address this challenge, techniques such as QXtract [2], PRDualRank [12], FactCrawl [8], and RSVM-IE and BAgg-IE [6] learn a collection-specific set of text queries (e.g., consisting of words and phrases) to target the useful documents and ignore the rest, thus reducing the overall extraction cost drastically. These techniques, however, do not attempt to predict which text collections are useful for the IE task—and hence merit further processing—and which ones will not contribute any useful output—and hence should be ignored, for efficiency.

In this paper, we introduce and address the problem of ranking deep web collections for an IE task, to prioritize the extraction effort by focusing on collections with substantial numbers of useful documents for the IE task. An approach for this task should rightfully conclude, for example, that FEMA is better for extracting *Occurs-in* tuples than PubMed. This collection ranking problem is related to the problem of resource selection in distributed information retrieval [29, Chapter 3], to identify topically relevant collections for a given user query. Unlike in distributed IR, though, our IE scenario requires that we identify collections

CIKM '15 Melbourne, Australia

DOI: http://dx.doi.org/10.1145/2806416.2806581.

[1] http://www.fema.gov/
[2] http://www.ncbi.nlm.nih.gov/pubmed

with useful documents for the IE task, rather than documents that are topically relevant for a given query. Despite this difference in focus, we can adapt resource selection approaches to our IE scenario, as we will see, as well as develop alternative, IE-specific approaches.

To prioritize deep web collections for an IE task, we must identify the most useful collections, namely, the collections with the largest numbers of useful documents for the IE task. Therefore, we need to estimate the number of useful documents in each collection and, importantly, we need to do so efficiently (e.g., by issuing a relatively small number of queries to each collection). For this estimation problem, we could exploit state-of-the-art techniques for measuring certain deep web collection properties (e.g., their number of documents) [4, 34, 35]. Unfortunately, as we will see, such techniques can be prohibitively expensive for IE, because they may need to issue many queries to sufficiently cover the (often rare) useful documents for an IE task of interest.

To address the limitations of generic estimation techniques, and to effectively rank deep web collections for a given IE task, we develop approaches that target the useful documents for the IE task in question. We compare both (adaptations of) state-of-the-art resource selection strategies and IE-specific approaches in an extensive experimental evaluation over realistic deep web collections, and for several different IE tasks. In summary, our key contributions are:

- We introduce the problem of ranking deep web collections for efficient and scalable IE (Section 2).
- We study traditional as well as IE-specific approaches for estimating, for each deep web collection, the number of useful documents for a given IE task (Section 3).
- We report the results of an extensive evaluation of both (adaptations of) traditional approaches for distributed information retrieval and IE-specific approaches over real-world deep web collections and for several different IE tasks. Our results show the merits and limitations of the alternative families of approaches, and provide a roadmap for addressing this critically important building block for efficient, scalable information extraction (Sections 4 and 5).

2. PROBLEM DEFINITION

Our focus is on the efficiency and scalability of the IE process over deep web text collections, whose contents can only be accessed via querying and cannot be retrieved using traditional Web "crawlers." To run an IE system over the available deep web collections, a naïve, expensive approach could resort to state-of-the-art approaches for efficient query-based IE execution (e.g., [2, 6, 8, 12]) over each deep web collection individually.[3] Such a naïve approach would be unnecessarily expensive, because not all collections contain any *useful* documents, or documents from which the IE system at hand manages to extract tuples. Therefore, to prioritize the IE effort, for efficiency, we focus on the problem of ranking deep web collections for an IE task of interest.

A related problem, *resource selection*, has been studied in the context of distributed information retrieval, to rank collections according to their topical relevance to a given user query [29]. Resource selection approaches generally consist of two steps: (1) *descriptor generation:* in an offline step, build a compact, representative collection summary (e.g., consisting of word frequency vectors [11, 15] or document samples [30,32]); (2) *relevance estimation:* to process a given query, use the collection descriptors to estimate the number of topically relevant documents in each collection, and rank the collections accordingly.

[3]Our approach is not applicable over open information extraction scenarios (e.g., [3]) where documents frequently contribute tuples to the open-ended extraction task.

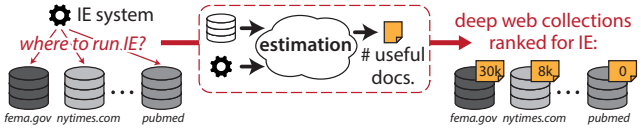


Figure 1: Collection ranking for IE over the deep web.

Unlike in distributed IR, our IE scenario requires that we identify collections with useful documents for the IE task, rather than collections with documents that are topically relevant to a given query. As a result, the collection descriptors will need to effectively capture the characteristics of the useful documents, a challenging proposition because of two critical reasons. First, the notion of document usefulness is, by definition, specific to a given IE task, so our collection ranking approaches—and the collection descriptors on which they rely—will have to be flexible to adapt to each given IE task. In particular, the "one-size-fits-all" descriptors adopted by resource selection approaches for distributed IR would not be appropriate for our IE scenario. Second, the fraction of documents in a collection that are useful for an IE task can many times be very small, so our collection ranking approaches—and the estimation techniques on which they rely—will have to effectively target the useful documents, to keep the ranking overhead to manageable levels. In particular, state-of-the-art techniques for estimating certain deep web collection properties [4,34,35], as discussed above, would be prohibitively expensive for our IE scenario.

We summarize the problem that we address as follows:

PROBLEM DEFINITION 1. *Consider a set of deep web collections and an information extraction task T with its corresponding (previously trained) information extraction system. Our goal is to rank the collections according to their number of useful documents for the IE task T (see Figure 1). Furthermore, the ranking process should be efficient (e.g., in terms of the number of queries issued to each collection), to keep its overhead to reasonable levels.*

Earlier efforts to identify collections for an extraction task (e.g., [1, 21]) have focused on examining the quality of the extraction output, rather than its volume. The (complementary) methods described in this paper can be adapted to consider quality (see Section 6).

3. ESTIMATING COLLECTION USEFULNESS

To prioritize the IE effort and rank deep web collections for an IE task, we need to estimate the number of useful documents for the IE task in each collection. Specifically, for each collection C, we will estimate the cardinality of C^{U}, the set of useful documents in C for the IE task at hand. In this section, we first provide an overview of three families of state-of-the-art estimation approaches that we can adapt for the task (Section 3.1) and then describe each method in detail (Sections 3.2 through 3.4).

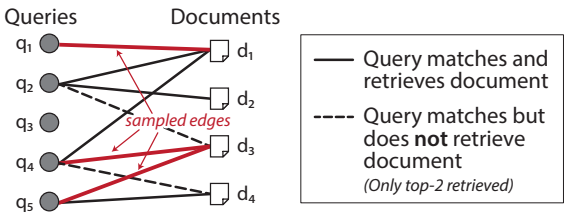


Figure 2: An example (query, document)-graph: the estimate contribution $f(d_1)$ from document d_1 has a $\frac{1}{3}$ weight (since its sampled degree is 3) and $f(d_3)$ is counted twice, each time with weight $\frac{1}{2}$.

Family	Technique (B or IE)	Collection	IE	
Surrogate	ReDDE (B)	Document sample + size estimate	Queries	
	Surrogate (IE)	Term-frequency map + size estimate	Document sample	
Query Pool-Based	PB (B)	In sum: -	In generic: -	In specific: query pool
		In avg: size estimate		
	PB-W (IE)	In sum: -	In generic: weights	In specific: query pool + weights
		In avg: size estimate		
Query Pool-Free	PF (B)	Query sample	-	
	PF-W (IE)	Query sample	Query sample	

Table 1: Summary of the characteristics of the baseline and IE-specific methods of Section 3.

3.1 Overview of Estimation Approaches

Given an IE task, we can cast the problem of estimating the number of useful documents for the task in a deep web collection as an instance of the generic task of estimating a "property" of interest for a deep web collection, which has been studied extensively in the research literature. Such a property F of a collection C is typically defined as an aggregate over a document-level function, say, $f(d)$. A simple example is the estimation of the number of documents in a deep web collection C: in this case, $f(d) = 1$ for every document d in C, and $F(C) = \sum_{d \in C} f(d)$. In our IE scenario, to estimate the number of useful documents we should define $f(d) = \mathbb{1}\{d \text{ is useful}\}$, that is, as the indicator function that returns 1 if d is useful and 0 otherwise. Various methods have been proposed to estimate properties of (queryable) document collections (e.g., collection size, number of documents relevant to a query, average document length) [4, 19, 34, 35], and these methods can be classified in three broad classes: (i) surrogate-based methods, (ii) query pool-based methods, and (iii) query pool-free methods.

Surrogate-based methods construct an approximate representation of the entire collection, and then use that surrogate to estimate the metric of interest, without further accessing the actual collection. A surrogate typically comprises a (relatively small) document sample (e.g., [30, 32]), or document frequency estimations for the terms occurring in the collection (e.g., [11, 15]). In resource selection for distributed IR, such representations have been widely used to estimate the number of documents relevant to a query (e.g., CORI [11], GlOSS [15], ReDDE [32], and Relax [30]). For example, ReDDE builds a random document sample S for each collection C, once and for all. Simply stated, to judge the relevance of C to a query q, ReDDE extrapolates the number of documents relevant to q in S to the entire collection.

Query pool-based methods pick queries from a predefined query pool Q (e.g., a dictionary of words or n-grams collected from extensive web crawls) to retrieve—from the collection at hand—documents from which to estimate the metric of interest. Unlike the collection-specific surrogates, the query pool can be shared across collections and can also be targeted specifically to the estimation task at hand. The query pool-based method in [23] aims at estimating collection size effectively but is inefficient: for large collections, the samples required to produce accurate estimates are very large. Subsequent (query pool-based) methods addressed this limitation, and sample random edges from a (query, document)-graph, as sketched in Figure 2: the graph vertices are queries $q \in Q$ and documents $d \in C$, and a solid edge (q, d) means that q retrieves d. For each sampled (q, d) pair, the measure $f(d)$ contributes to the estimation of $F(C)$ with a weight that is proportional to the probability of having sampled d (see Section 3.3). Pool-based methods work well, provided the query pool Q retrieves all documents of interest for the metric (i.e., for which $f(d) \neq 0$).

Query pool-free methods avoid relying on a query pool, and rather find the queries to issue "on the fly." These methods use a seed query (e.g., a common word or phrase) to retrieve a first set of documents and select the next query to issue from these documents. Thus, they issue queries and retrieve documents to perform a random walk on a graph where nodes are either queries or documents. To properly weigh the particular $f(d)$ value derived from visiting a node in such graph, these methods use the fact that the probability of visiting a node during a random walk is proportional to its number of incident edges.

In our IE scenario, we need to estimate the number of useful documents in a deep web collection for an IE task of interest. Unfortunately, only very few or no useful documents might be present in a truly random document sample from the collection, given that useful documents might be rare for an IE task. This poses a problem for all three families of estimation methods summarized above. The next subsections describe the existing approaches in detail and derive IE-specific estimators that effectively target useful documents. By aiming to collect documents for which $f(d) \neq 0$, our IE-specific methods are designed to have more non-zero terms in their estimation, thus alleviating the limitations of existing estimation techniques for other tasks. Each subsection first discusses a state-of-the-art baseline method and how we can apply it to our IE scenario; the second part introduces an IE-specific method, designed specifically for collection usefulness estimation. Table 1 summarizes the requirements of each method to handle collections and IE tasks.

3.2 Surrogate-Based Estimators

Baseline (ReDDE): Our first (baseline) estimator is an adaptation of ReDDE, a resource selection technique for distributed IR [31]. To estimate the number of topically relevant documents for a query q in a collection C, ReDDE predicts the relevance of a representative sample $S \subset C$ and scales it to the entire collection with a factor $SF = |C|/|S|$ to obtain a collection relevance metric $\text{Rel}(q, C)$. This metric relies on estimating (i) the collection size $|C|$ and (ii) the relevance of sample documents in S to the given query q. The size $|C|$ is query-independent and thus computed once and for all, while the query relevance for d is based on issuing q to a centralized global sample unifying all individual collec-

tion samples. To apply ReDDE to our IE scenario, we model the IE task as a set of high-performing queries Q_{IE}, which we can automatically learn (e.g., see QXtract [2]). We thus calculate collection relevance for the IE task as a sum of the ReDDE relevance metric, taken over the top-k queries from our set of IE-specific queries: $|\hat{C^{\text{U}}}| = \sum_{q \in \text{top}_k(Q_{\text{IE}})} \text{Rel}(q, C)$.

IE-specific (Surrogate): An alternative, IE-specific estimation approach is to collect a random sample S from a collection C, run the IE system over the documents in S, and extrapolate the number of useful documents to the full collection C as in ReDDE, namely, $|\hat{C^{\text{U}}}| = SF \cdot |S^{\text{U}}|$ with scaling factor $SF = |C|/|S|$. Unfortunately, a random sample might have very few or no useful documents because, as discussed, useful documents for an IE task are often rare. To address this problem, the document sample S should be then biased towards useful documents. This implies that we subsequently have to correct the scaling factor SF for that usefulness bias of S. The main idea is to look at document frequency differences of certain terms between the sample S and the full collection C. For a given term t, let $df(t, X)$ be the fraction of documents in X that contain t, and define the frequency ratio in the sample vs. in the full collection as $\alpha_t^S \triangleq \frac{df(t,S)}{df(t,C)}$. We propose to use the average of this ratio over *useful terms*, T^{U}, as a heuristic scaling factor: $SF = \mathbb{E}_{T^{\text{U}}}\left[1/\alpha_t^S\right]$, where useful terms are those that are (i) biased towards usefulness, in contrast to "neutral" terms that would appear equally in useful and useless documents; and (ii) overrepresented in the sample (i.e., $\alpha_t^S > 1$). The rationale for this choice of SF is that the overrepresentation of such useful terms in S compared to in C is a good proxy for the overrepresentation of useful documents. (We can find T^{U} through standard statistical significance tests.) Importantly, we assume that we know the document frequencies $df(t, C)$ in the complete collection, as well as the collection size $|C|$. We can estimate these values reliably once and for all for each collection (e.g., see [20] and [31]).

3.3 Query Pool-Based Estimators

Baseline (PB): We consider the method of Bar-Yossef *et al.* [4], which set the foundations for subsequent query pool-based estimators (e.g., [34]) and allows estimating any generic metric that can be expressed as a discrete integral over the documents in C: $\text{Int}_\pi(f) \triangleq \sum_{d \in C} f(d) \cdot \pi_D(d)$, where f is the target function of interest and π_D weighs documents as needed (e.g., $\pi_D(d) = 1$ if all documents contribute equally). As indicated in Section 3.1, documents $d \in C$ are obtained by issuing queries from a pool Q. Bar-Yossef *et al.*'s method relies on two core ideas: (i) extend π_D to a measure over the (query, document)-space $Q \times C$, and (ii) apply importance sampling to use a practical sampling strategy, selecting a (query, document)-pair with probability $p(q, d)$, rather than from the probability distribution induced by π_D, which is unfeasible to sample from. For (i), the extended measure is:

$$\pi(q, d) \triangleq \frac{\mathbb{1}\{d \in C_q\} \cdot \pi_D(d)}{\omega(d)},$$

where C_q is the set of documents that q retrieves from C and $\omega(d)$ is the degree of document d, defined as the number of queries in Q that retrieve d. For (ii), the practical sampling strategy is: (1) pick a random query q from the set of queries that return at least one document, noted as Q_+, and then (2) randomly pick one of the documents it retrieves:

$$p(q, d) \triangleq \frac{1}{|Q_+|} \cdot \frac{\mathbb{1}\{d \in C_q\}}{|C_q|}.$$

In importance sampling, $p(q, d)$ and $\pi(q, d)$ induce probability distributions referred to as the *trial* and *target* distributions, respectively, and define the importance weight as:

$$w(q, d) \triangleq \frac{\pi(q, d)}{p(q, d)} = \frac{\pi_D(d) \cdot |Q_+| \cdot |C_q|}{\omega(d)}.$$

Bar-Yossef *et al.* use an efficient estimator $u(q, d)$ of $w(q, d)$, defined as $u(q, d) = \pi_D(d) \cdot \text{PSE} \cdot |C_q| \cdot \text{IDE}(d)$, with a pool size estimator PSE for $|Q_+|$ and an inverse degree estimator IDE(d) for $1/\omega(d)$, in turn, to calculate the approximate importance sampler (AIS): $\text{AIS}(q, d) \triangleq f(d) \cdot u(q, d)$. The authors show that if u approximates w well, and if the ratio u/w is uncorrelated with f, AIS remains largely unbiased.

A variant of the above estimator for sums [4] computes $\text{Avg}(f) = \sum_i (f(d) \cdot u(q_i, d_i)) / \sum_i u(q_i, d_i)$ (where the PSE factor cancels out) and obtains the estimator by multiplying Avg(f) by the size of the collection. This collection size estimation can be done once and for all, and reused for different metrics (e.g., for the usefulness for different IE tasks), to amortize its cost.

However, directly using AIS in our IE scenario is problematic: (i) the number of queries to issue and subsequent IE-processing of returned documents to determine $f(d)$, to find some useful documents, may be high; and (ii) the estimation will have high variance because of the few non-zero $f(d)$ values.[4] To address the first limitation for the related problem of counting the frequency of a given word (e.g., "sports") in a collection, Zhang *et al.* identify the queries that are positively correlated with the word (e.g., query "golf") [34]. The query sampling process is then stratified over correlated and uncorrelated queries. Unfortunately, this approach still requires issuing a large number of queries.

IE-specific (PB-W): We adapt the Bar-Yossef *et al.* approach, with target and trial distributions that are aligned with $f(d)$ for usefulness: Our *target distribution* assigns probabilities greater than 0 only to useful documents:

$$\pi_{\text{U}}(q, d) \triangleq \frac{\mathbb{1}\{d \in C_q^{\text{U}}\} \cdot \pi_D(d)}{\omega(d)},$$

where C_q^{U} is the set of useful documents that q retrieves from C. Our *trial distribution*, accordingly, should (i) retrieve useful documents with high recall and precision, and (ii) be efficient to sample from. Specifically, for recall, we learn queries from a large, external text collection E that we can process once and for all. For precision, we learn the usefulness of queries with respect to the given IE task: (1) process E with the IE system at hand, (2) query E with all words in the documents, and (3) count the useful and useless documents within the top-k results for each query q, noted as $|E_q^{\text{U}}|$ and $|E_q^{\text{N}}|$, respectively. Finally, our trial distribution assigns a selection weight to each query q, defined as $w \cdot |E_q^{\text{U}}| + |E_q^{\text{N}}|$.[5]

Given these query weights, our trial process consists of two steps: (1) pick a useful query q proportionally to its weight,

[4]The variance can be reduced via Rao-Blackwellization, as suggested in [4], which requires running IE over all retrieved documents. We evaluate this version of the algorithm later in the experimental section.

[5]$|E_q^{\text{N}}|$ in the selection weight operates as a smoothing factor.

and (2) pick a document from q's useful results uniformly at random. This yields the following trial distribution:

$$p_{\text{U}}(q,d) \triangleq \frac{w \cdot |E_q^{\text{U}}| + |E_q^{\text{N}}|}{\mathbb{Z}_w} \cdot \frac{\mathbb{1}\{d \in C_q^{\text{U}}\}}{|C_q^{\text{U}}|}.$$

Here, $\mathbb{Z}_w = \sum_{q \in Q_+^{\text{U}}} w \cdot |E_q^{\text{U}}| + |E_q^{\text{N}}|$ is the normalization factor of the probability distribution induced by the queries that retrieve at least one useful document, Q_+^{U}. We can now obtain our importance weight function as:

$$w_{\text{U}}(q,d) \triangleq \frac{\pi_{\text{U}}(q,d)}{p_{\text{U}}(q,d)} = \frac{\mathbb{Z}_w \cdot \pi_D(d)}{\omega(d)} \cdot \frac{|C_q^{\text{U}}|}{w \cdot |E_q^{\text{U}}| + |E_q^{\text{N}}|},$$

which we need to compute only over the useful documents. Similarly to [4], we rely on an efficient estimator u_{U} of w_{U}, defined as:

$$u_{\text{U}}(d,q) \triangleq \frac{\text{NFE} \cdot \text{IDE}(d) \cdot \text{UE}(q) \cdot \pi_D(d)}{w \cdot |E_q^{\text{U}}| + |E_q^{\text{N}}|},$$

to keep estimation costs to reasonable levels. Here, NFE and UE(q) are estimators of normalization factor $\mathbb{Z}_w$ and number of useful documents retrieved by a query.

The key challenge in computing NFE is to account for the number of queries $|Q_+^{\text{U}}|$ and the distribution of their selection weight. We can compute them both on the fly while sampling during our trial process: Both factors can be computed by sampling according to the selection weight, instead of uniformly at random, as in Bar-Yossef *et al.*'s PSE. We compute NFE by keeping track of the fraction of sampled queries that retrieve at least one useful document, defined as α, and computing $\alpha \cdot \sum_{q \in Q}(w \cdot |E_q^{\text{U}}| + |E_q^{\text{N}}|)$.[6] To compute UE($q$), we proceed similarly to Bar-Yossef *et al.*'s IDE computation: we sample documents uniformly at random from C_q until we find a useful one; if we find the useful document after processing n documents, then $\text{UE}(q) = \frac{|C_q|}{n}$.

3.4 Query Pool-Free Estimators

Baseline (PF): We focus on the method introduced by Zhang *et al.* [35], using a query graph: (i) the nodes are h-grams[7] q that retrieve at least u documents[8] and (ii) undirected edges connect a node pair (q, q') if q' matches (i.e., appears in the text of) at least one of the documents that q retrieves, and vice versa. Since a random walk implies that a node q is visited with a probability proportional to its degree $d(q)$, each per-query estimate is weighted with $1/d(q)$ to agree with uniform sampling. The estimation of a function $F(C) = \sum_{d \in C} f(d)$ from sampled queries S in (query) graph Q is:

$$\hat{F}(C) = |\hat{Q}| \cdot \underbrace{\frac{\sum_{q \in S} 1/d(q) \cdot \sum_{d \in C_q} f(d)/\omega(d)}{\sum_{q \in S} 1/d(q)}}_{\#\text{ useful documents per sampled query}}$$

$$= |V_c| \cdot \tilde{\lambda}(V_c) \cdot \frac{\sum_{q \in S} 1/d(q) \cdot \sum_{d \in C_q} f(d)/\omega(d)}{\sum_{q \in S \cap V_C} 1/d(q)}. \quad (1)$$

Here, $\omega(d)$ is the number of queries in Q that retrieve document d. Furthermore, the size of the query graph $|\hat{Q}|$ is estimated from a startup query collection V_C, which is a sample (e.g., obtained from the documents of another random walk, independent of S) of the vocabulary of h-grams appearing in the collection, and its estimated fraction $\tilde{\lambda}(V_C)$ that retrieve at least u documents when issued as a query.[9] While eliminating a potential coverage issue by avoiding an a priori query pool, the resulting estimation may still require many queries in the IE scenario, to find sufficiently many useful documents.

IE-specific (PF-W): To estimate the number of useful documents, we could restrict the graph to only useful queries (i.e., queries that retrieve at least one useful document and for which $f(d) = 1$). However, such graph could be largely disconnected and the random walk would be unable to fully explore it. Instead, we keep the original graph and modify the random walk process to favor visiting useful queries: We define a *weighted graph*, on which we perform a "weighted" random walk (i.e., edge e with weight $w(e)$ is selected from an edge set E with probability $w(e)/\sum_{e' \in E} w(e')$). We define an edge (q, q') to be *useful* if and only if both queries it connects retrieve at least one useful document. We assign useful edges weight w and the others 1. In Equation (1) we thus replace the original (unweighted) degree $d(q)$ with the weighted counterpart: $d_w(q) = w \cdot N_{\text{U}} + N_{\text{N}}$, where N_{U} is the number of q's useful incident edges (i.e., useful neighbors q') and $N_{\text{N}} = N_{\text{ALL}} - N_{\text{U}}$ (with $N_{\text{ALL}} = d(q)$). The definition of $\omega(d)$ (i.e., the number of queries that retrieve it) remains unchanged, thus we still estimate it using the method in [35].

To estimate the weighted degree $d_w(q)$ of a sampled query q we need (i) the number of all incident edges N_{ALL}, and (ii) the number of useful edges N_{U}. For N_{ALL} we proceed as in [35]. For N_{U} we proceed similarly, now counting the number of sampling attempts n_{U} we need to find a q' that both matches q and is useful, to estimate $\hat{N}_{\text{U}} = |Q'|/n_{\text{U}}$.[10] Thus, we calculate $\hat{d}_w(q) = w \cdot \hat{N}_{\text{U}} + (\hat{N}_{\text{ALL}} - \hat{N}_{\text{U}})$.

In this section, we described the baseline and IE-specific approaches that we study in this paper. We now describe the settings for our experimental evaluation of these techniques (Section 4) and report our results (Section 5).

4. EXPERIMENTAL SETTINGS

Collections: Our *test set* consists of 96 real web collections across different topics, collected using an approach similar to that in [16] over the Open Directory Project (ODP)[11]. Specifically, we first selected the 8 top-level ODP categories with the largest number of entries, namely, Business, Society, Arts, Science, Computers, Recreation, Shopping, and Sports. We then selected the 5 most popular subcategories in each of the 8 initial categories. In turn, we also picked the 5 most popular subsubcategories from each subcategory, for a total of 200 subsubcategories. For each subsubcategory, we then randomly chose 7 unique web collections that have a text search interface. Finally, we collected their contents using a state-of-the-art query-based sampler [10], issuing at most 20,000 queries and retrieving up to 1,000 documents for each. In our test set, we kept the collections that produced

[6] The $|E_q^{\text{U}}|$ and $|E_q^{\text{N}}|$ values in this formula are computed once and for all during the generation of the queries.

[7] Zhang *et al.* argue that $h = 1$ works well in practice.

[8] Parameter u controls the size and connectivity of the graph. Zhang *et al.* propose $u = 3$.

[9] We correct an erroneous $1/|S|$ factor from [35, eq. (5)].

[10] Implementation-wise, we can get n_{ALL}, the sampling attempts for N_{ALL}, from the same sampling sequence as n_{U}.

[11] `http://www.dmoz.org/`

Relation	Useful documents	
	SSK	BONG
Person–Career	56.20%	55.95%
Natural Disaster–Location	2.03%	2.74%
Man Made Disaster–Location (*)	0.80%	0.87%
Person–Charge	1.55%	1.84%
Election–Winner	0.24%	0.84%

Table 2: Fraction of useful documents found in the TREC 1-5 collections for relations extracted using two IE systems, SSK and BONG. We use (*) only during tuning.

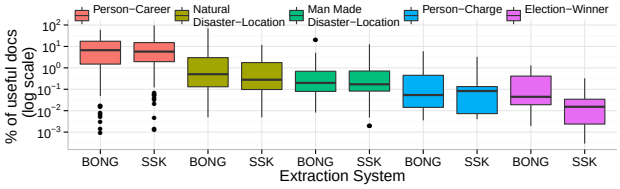


Figure 3: Fraction of useful documents for each relation across our 96 test collections. The box boundaries are the 25th and 75th percentiles, the bold horizontal line inside each box is the median, and the dots denote outliers.

at least 10,000 documents following this method, to focus the evaluation on collections with a substantial number of documents. Our *tuning set*, which we use for tuning parameters of the various techniques, consists of 40 collections selected randomly from among the collections under the above subsubcategories and not in the test set. We collected documents from these tuning collections using the Nutch Web crawler[12]. We indexed each collection, in both the tuning set and the test set, with the text retrieval toolkit Lucene[13], to emulate the query-only behavior of deep web collections and only access them through their query interface. We exhaustively processed the collections with our IE systems (see below) to obtain the real number of useful documents in each collection. We also used TREC 1-5 collections[14] for different operations (e.g., query pool construction), which we describe as needed throughout this section.

Information Extraction Systems: We evaluated a variety of IE systems and components for all relations in our experiments (see below) via 5-fold crossvalidation over a set of training documents, and selected the two best-performing combinations, namely, Subsequence Kernel (SSK) [9] and Bag of n-Grams (BONG) [14]. We implemented them using REEL[15] [5]. We also considered different named entity taggers and selected: for *person* and *location* entities, the pretrained conditional random fields (CRF) [26] from the StanfordNLP package[16]; for *natural disasters*, CRFs from the etxt2db framework[17]; for the remaining entities, maximum entropy markov models (MEMM) [25], also from etxt2db.

Relations: For a robust evaluation, we include 5 substantially different relations in the experiments (see Table 2). Four such relations, namely, Natural Disaster–Location, Man Made Disaster–Location, Person–Charge, and Election–Winner, are sparse, in that very few documents tend to be useful for them. In contrast, relation Person–Career is a dense relation. (Table 2 shows the percentage of useful documents for each relation in the TREC 1-5 collections for the two IE systems. Also, Figure 3 shows the distribution of useful documents over the collections in our test set.)

Technique Tuning: We tuned each technique over the tuning set and using the SSK IE system over Man Made Disaster–Location.

Surrogate-based methods (Section 3.2): (1) Baseline: We evaluated different query sets Q and numbers of queries k. For Q, we generated one-word queries using the SVM approach in [2] and two effective feature selection methods, namely, Information Gain and χ^2 test [36], over 10,000 documents (50% useful and 50% useless) from TREC 1-5; we kept the words that are discriminative of the useful class; also, and as suggested in [2], we evaluated a query set including all words in the documents and another one removing the tuple attribute values. We varied $k \in [10, 200]$ in intervals of 10. We built the document samples using the query-based sampling technique in [10], which produces nearly-random document samples from words collected from retrieved documents. *(2) IE-specific:* We compared (i) the query-based sampling technique in ReDDE; (ii) the bootstrapping-based sampling approach in [2], which starts from a set of seed tuples as queries that is iteratively expanded as it collects more useful documents and extracts tuples from them; (iii) a sampling technique that issues the queries in Q above based on their score. Finally, to find the useful terms and to asses term bias we used the Fisher exact test in [13] varying p-value $\in [0.01, 0.1]$.

Query pool-based methods (Section 3.3): We built two query pools of single words from TREC 1-5: (i) a generic query pool (G) of 4M words considering all documents; and (ii) an IE specific query pool (S) that only considers words from useful documents. We performed the estimation process with and without Rao-Blackwellization, suggested in the original paper for variance reduction [4]. We evaluated the sum and average approaches. For the IE-specific method, we also: (i) evaluated several different values for the weight $w \in [50, 500000]$; (ii) used the TREC 1-5 collections as our external collection; and (iii) varied the number of documents to retrieve $k \in [10, 1000]$ for weight computation.

Query pool-free methods (Section 3.4): We varied three parameters common to the baseline and IE-specific approach: (i) the number of sampled queries $|S| \in [10, 500]$, with increments of 10, (ii) the length of h-grams, $h \in \{1, 2, 3\}$, and (iii) the length of the burn-in of the random walk b before collecting the samples, $b \in [50, 500]$ with increments of 50. For the IE-specific method, we evaluated several different values for its weight $w \in [10, 5000]$.

Techniques for Comparison: We compared the following alternatives over the 96 collections in our test set, with the settings derived via tuning, as summarized below:

- *Baseline surrogate-based (ReDDE):* We generate the samples with the query-based document sampler in [10], issuing up to 300 queries and collecting (at most) 5 documents for each. We use the queries learned with χ^2 for Q and use the top-100 queries (i.e., $k = 100$).
- *IE-specific surrogate-based method (Surrogate):* For sampling, we derive Q using the χ^2 method and excluding tuple attributes. We considered the useful queries in order of χ^2.

[12] http://nutch.apache.org/
[13] http://lucene.apache.org/
[14] http://trec.nist.gov/data.html
[15] http://reel.cs.columbia.edu/
[16] http://nlp.stanford.edu/software/CRF-NER.shtml
[17] http://web.ist.utl.pt/rui.lageira/

Our sample S has at most 1000 documents. To select the useful terms $T^{\cup}$, we use p-value = 0.05 in the Fisher test.

• *Baseline query-pool-based (PB):* We use the sum (ABS) and average (AVG) methods using the G and Q query pools and applying Rao-Blackwellization. (The impact on efficiency of applying Rao-Blackwellization is low, because the number of documents processed with the IE system is small.)

• *IE-specific query-pool-based method (PB-W):* As with the baseline, we evaluate sum (ABS) and average (AVG) estimators using the G and S query pools and applying Rao-Blackwellization. We index our external collection using Lucene with default parameters. We use $k = 1000$ for the query weight computation and $w = 5000$.

• *Baseline query-pool-free (PF):* We use an English dictionary to randomly find an initial query for the estimation process. We use single terms as queries (i.e., $h = 1$) and only accept queries that retrieve at least $u = 3$ documents, as suggested in [35]. We collect 100 queries for V_c, the startup query path that can be shared across IE tasks, and at most 100 for the estimation sample queries S.

• *IE-specific query-pool-free method (PF-W):* We use a similar configuration to that in PF, but with $w = 1000$.

Additional Settings: Estimators issue at most 100 queries, and retrieve up to 50 documents per query. To account for randomness, we run each estimator five times and report average values over the five runs. Finally, note that all estimators contain some form of (weighted) averaging of a metric over samples S (e.g., for the pool-free estimator, we calculate the individual contribution of each q in the summation in the denominator of Equation (1)). We filter outliers from this average, using the outlier detection algorithm in [33, "Method I"] as implemented in the R[18] package "extremevalues": a value x is an outlier if it is outside the limit where less than 1 observation is expected, based on observed data within quantile limits $[\alpha, 1-\alpha]$. We evaluated every estimator with and without outlier removal (using $\alpha \in \{0.05, 0.1\}$). Removing outliers using $\alpha = 0.1$ performed best across all techniques. Thus, our final results include such removal.

Evaluation Metrics: We measure *ranking quality* and *estimation cost* with the following metrics:

• *Cumulative Gain (CG@k):* We measure the number of useful documents that we obtain by processing the collections in ranking order. If u_i is the number of useful documents in the ith collection, we define $\text{CG@}k = \sum_{i=1}^{k} u_i$. This metric focuses on absolute values of useful documents, which makes the comparison across relations problematic, and also does not fully capture "errors" in the ranking.

• *Normalized Discounted Cumulative Gain (nDCG@k):* nDCG@k alleviates the limitations above in a robust manner. Specifically, nDCG@k is defined as the normalized version of Discounted Cumulative Gain $\text{DCG@}k = u_1 + \sum_{i=2}^{k} \frac{u_i}{log_2(i)}$, which penalizes the errors in the ranking order.[19] Now, to normalize DCG@k—and obtain nDCG@k—, we need to calculate the DCG@k of an ideal ranking, namely, IDCG@k. Finally, $\text{nDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$.

• *Processed Documents (PD) and Issued Queries (IQ):* We measure the efficiency of the IE process in terms of the number of issued queries and processed documents, and not running time. The reasons for this are twofold: (1) many factors (e.g., network traffic, collection responsiveness) can distort running times in the distributed environments on which we focus and are difficult to capture reliably; and (2) the number of issued queries and processed documents are good indicators of expected running time. Finally, we report these values only for the actual estimation process and ignore the initial, once-and-for-all processing (e.g., collection size estimation or random document sample generation) required by the techniques, which gets amortized over time.

[18] `http://www.r-project.org/`

[19] Variants of DCG@k exist in the literature; the version we use accounts for the distribution of useful documents.

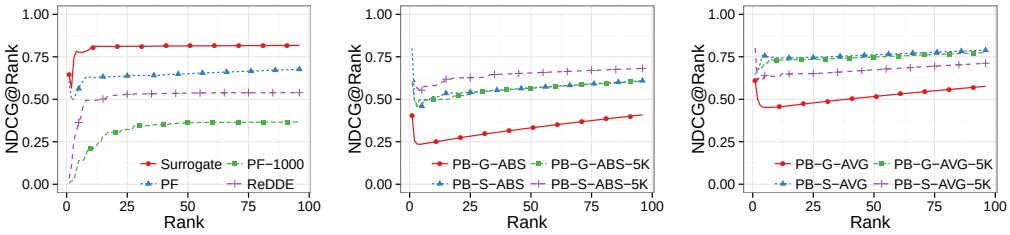
5. EXPERIMENTAL RESULTS

We now evaluate the baseline and IE-specific ranking approaches of Section 3, with the settings of Section 4.

Quality of collection ranking approaches: We evaluate the ranking approaches over all relations and IE systems of Section 4. Figure 4 shows nDCG@k of all techniques over the entire rank of collections (i.e., $k \in [1, 96]$) for Natural Disaster–Location using the BONG IE system, and issuing at most 100 queries. (Other relations and IE systems—with the exception of Person–Career, which we study in detail later—yielded similar results. Also, we later vary the number of issued queries.) Figure 4a, for the surrogate and query pool-based methods, shows that the PF baseline outperforms PF-1000, its IE-specific counterpart, by almost 75%. PF-1000 requires on average more queries than PF to walk the random graph; for this reason, PF-1000 will rarely find useful documents at such small query budget. As we will see, when PF-1000 finds useful documents, its performance improves considerably, always overcoming its baseline counterpart. In contrast, among the surrogate methods, Surrogate outperforms ReDDE by almost 50%: the IE-specific document sample, although small, manages to include useful documents; also, ReDDE's collection descriptors do not accurately characterize the useful documents.

Figures 4b and 4c, for the query pool-based methods for sums and averages, show that the IE-specific versions also outperform the baseline counterparts. For sum, this difference is mainly based on the number of useful documents sampled during the estimation, because there are more nonzero components to include in the estimation. For this reason, PB-S-ABS-5K is best, with its weighted specific query pool highlighting potentially useful queries. For average, the quality of the ranking also depends on finding queries that retrieve a combination of useful and useless documents, as both types are crucial for computing the average in question. PB-S-AVG and PB-G-AVG-5K sample such queries and thus exhibit the highest quality in this family. Overall, and across families, the top contenders are Surrogate and the average pool-based estimators, which effectively exploit both useful and useless documents during estimation.

To understand the actual number of useful documents observed as we process collections in ranking order, Figure 5 shows CG@k for Natural Disaster–Location and Person Career using the BONG IE system, for a selection of high-quality techniques according to our experiments. For reference we also show the CG@k of an ideal ranking (labeled Ideal) and of a random ranking (noted as a dashed line). Figure 5 reveals substantial gains from prioritizing the collections for extraction. For Natural Disaster–Location, for example, Surrogate effectively identifies the collections containing 95% of the total useful documents within the top-10 collections. This would translate to an efficiency improvement of almost 90% if we were to only process these top-



(a) Surrogate and pool-free (b) Pool-based (sum) (c) Pool-based (average)

Figure 4: nDCG@k for Natural Disaster–Location, for the BONG IE system and issuing (at most) 100 queries.

10 collections, because we would ignore 86 (out of 96) test collections. A noticeable benefit is also shown for Person–Career, which we discuss in detail later in this section, for Surrogate, ReDDE, and PB-G-AVG-5K.

Efficiency of collection ranking approaches: To understand the efficiency of the different approaches, we analyze the extraction cost and achieved ranking quality at different stages in the estimation process. Figures 6 and 7 show, respectively, the number of documents processed with the IE system and nDCG@10 for different numbers of issued queries, for Person–Charge using the SSK IE system. (Different relations and IE systems yielded analogous conclusions.) We show the same technique splits as in the above analysis, for clarity. Figure 6 shows that IE-specific techniques process on average more documents than their baseline counterpart. ReDDE is an exception: it does not incur querying or extraction costs during the estimation process.

The reasons as to why IE-specific approaches process more documents on average are manifold and differ for each family of techniques. Notably, for Surrogate (Figure 6a), the number of extracted documents grows with the size of the document sample, because all sampled documents need to be processed. For the pool-free family (also in Figure 6a), PF-1000 may need to process several documents before choosing the next "hop," due to its weighted walking strategy. In contrast, PF does not need this operation, since it navigates the graph only based on the document contents, without incurring additional extractions. However, these techniques may retrieve several hundred documents from each collection for graph construction, to extract h-grams from the documents.

Finally, for the pool-based families (Figures 6b and 6c), the extraction cost grows with the number of observed useful documents (from the sampled edges). In fact, after obtaining a useful document further operations take place: (i) for sum, the inverse degree estimator issues additional queries to the collection, although it avoids processing the documents with the IE system; the average estimators do not need inverse degree estimation and, therefore, more documents are often processed for the same query budget; and (ii) the Rao-Blackwellization method for variance reduction processes all documents that the current query retrieves; this increases the extraction cost, but this cost is comparable across the techniques we evaluated.

After analyzing the extraction cost, we also study the ranking quality associated with such costs. Figure 7 shows that the quality of the ranking improves with the number of issued queries: The estimators learn more—and more reliably—about the collections as the estimation progresses,

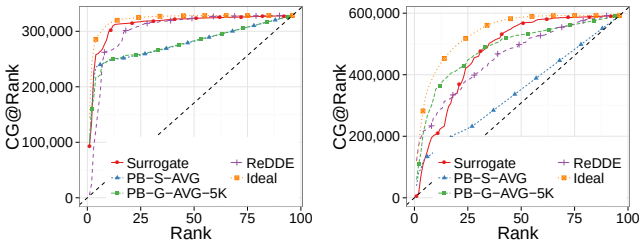
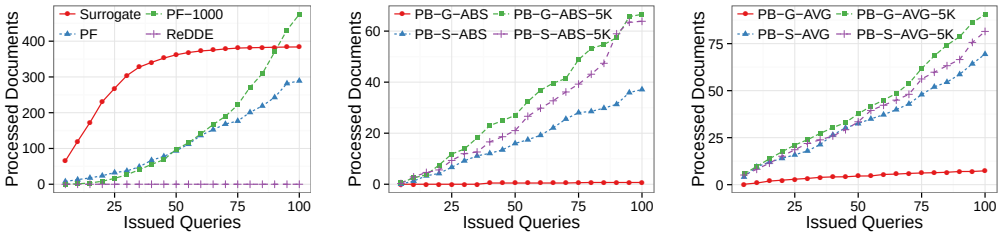


Figure 5: CG@k for Natural Disaster–Location (left) and Person–Career (right) for the BONG IE system and issuing (at most) 100 queries.

which is reflected in better estimations. These improvements, though, vary considerably across techniques: Surrogate, for instance, produces high-quality estimates even when issuing a small number of queries, whereas the remaining techniques improve their quality progressively, with a steeper gain for IE-specific than for baseline approaches. Furthermore, some IE-specific techniques yield higher quality rankings than other techniques at a fraction of their cost. For example, after issuing 25 queries, PB-G-ABS-5K and PB-G-AVG-5K exhibit comparable quality to PB-S-ABS and PB-S-AVG after issuing 100 queries, respectively.

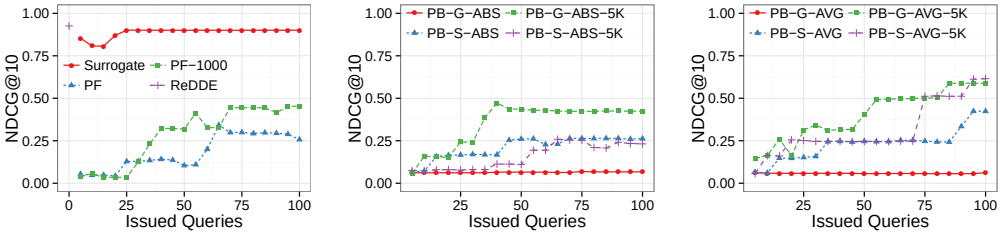
Impact of collection characteristics: Deep web collections are rather heterogeneous, with substantial differences in size and contents. Notably, large collections are problematic for baseline approaches, as Figure 5a shows for Natural Disaster–Location, where the most useful collections were among the largest collections (350,000 documents on average). We showed the effectiveness of our IE-specific approaches for this case. Similarly, the collection contents also affect some of the other approaches: Pool-based approaches retrieved substantially fewer documents than pool-free approaches (see Figure 6), because many queries in the query pool were not topically relevant to the contents of the collections; pool-free approaches do not exhibit this problem.

Impact of IE-task characteristics: To understand the impact of relation characteristics, we now focus on a dense relation. Figure 8 shows nDCG@k over the entire rank of collections (i.e., $k \in [1, 96]$) for Person–Career using the BONG IE system, and for all techniques. We show the same technique splits as in the above analysis, for clarity. Figure 8a shows that Surrogate, PF, and PF-1000 exhibit comparable (low) quality: these techniques failed to correctly



(a) Surrogate and pool-free (b) Pool-based (sum) (c) Pool-based (average)

Figure 6: Processed documents for Person–Charge, for the SSK IE system and different numbers of issued queries.



(a) Surrogate and pool-free (b) Pool-based (sum) (c) Pool-based (average)

Figure 7: nDCG@10 for Person–Charge, for the SSK IE system and different numbers of issued queries.

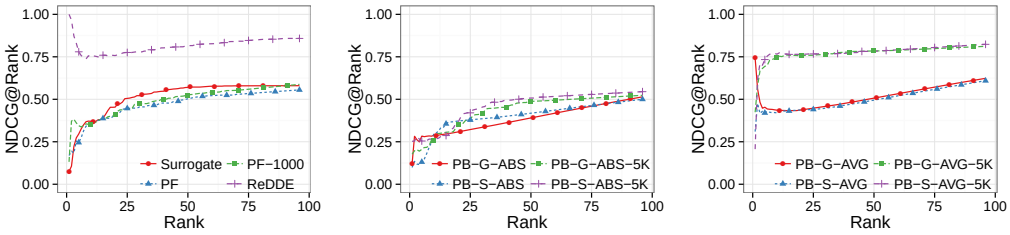
rank large collections with a large number—but a relatively low fraction—of useful documents. Specifically, Surrogate was unable to obtain discriminative terms for the estimation, since most sampled documents were useful. The pool-free approaches, on the other hand, were unable to reach many useful documents, because the useful documents in the top collections were a small fraction in each collection. The pool-based sum estimators in Figure 8b exhibit a trend similar to that of pool-free approaches. However, two sets of techniques performed relatively well, namely, ReDDE (Figure 8a) and the IE-specific pool-based AVG estimators (Figure 8c). These techniques benefited from the scaling to the entire collection, because the largest collections were among the top most useful collections.

Additional discussion: An orthogonal, interesting aspect to the discussion above concerns the overhead incurred by a ranking approach when new collections or IE tasks arrive. Table 1 summarized the collection- and IE-specific requirements of each ranking technique. For new collections, almost all techniques require some preprocessing (see collection column in Table 1). Deciding the approach to adopt will not only depend on the performance and overhead of the techniques but also on the number of IE systems that we will run over each (new) collection. Because size estimation and graph construction may take up to several thousand queries, approaches that rely on these will only be reasonable when many IE systems are involved, as the cost will amortize over time. In other cases, though, estimators that do not require additional information from the collection, such as pool-based estimators for sums, may be the best choice.

Similarly, for new IE tasks, almost all techniques also require some preprocessing (see IE column in Table 1). The most expensive process is, by far, producing a specific query pool (with or without query weights). The remaining processes, namely, learning queries for document sampling or for querying the descriptor in ReDDE and producing a query sample for the pool-free techniques, are relatively inexpensive. Therefore, and given our quality and efficiency results, surrogate methods (i.e., ReDDE or Surrogate) seem to be the most reasonable choice for new IE tasks. However, if new collections are expected to appear at high rates, it may be worthwhile building—and amortizing the construction of—a query pool, so that the estimation starts without overhead.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced and addressed the problem of ranking deep web collections for an IE task, to prioritize the extraction effort by focusing on collections with substantial numbers of useful documents for the IE task. Specifically, the problem is that of effectively and efficiently ranking a set of deep web collections according to their number of useful documents for a given IE task. We studied both (adaptations of) state-of-the-art resource selection strategies, and IE-specific approaches. We performed an extensive experimental evaluation over realistic deep web collections, and for several different IE tasks. Our evaluation focused on the quality and efficiency characteristics of the ranking approaches, with the following conclusions: (1) we found top contenders for each of these characteristics and provided insight on how to choose among them; (2) we analyzed which techniques are better suited for certain characteristics of the collections, such as their size and contents, and of the IE tasks, such as their sparsity; and (3) we discussed the overhead incurred by each technique in dynamic domains, where new collections or IE tasks may arrive. Overall, this paper provides a roadmap for addressing this critically important building block for efficient, scalable information extraction.



(a) Surrogate and pool-free (b) Pool-based (sum) (c) Pool-based (average)

Figure 8: nDCG@k for Person–Career, for the BONG IE system and issuing (at most) 100 queries.

As interesting future work, we will consider other factors for collection ranking, such as the quality, diversity, and novelty of the extraction output from the collections. Some of these factors (e.g., quality) just require defining an appropriate target measure f (e.g., as a function of the confidence scores of the tuples extracted from a document), as suggested in [4] and [35]. However, other factors (e.g., diversity and novelty) would also require analyzing the contents of multiple collections simultaneously, because the extraction output of one collection may be included in that of others.

Acknowledgments: This material is based upon work supported by the National Science Foundation under Grant IIS-08-11038 and by Bloomberg L.P. C. Develder was supported by a travel grant from the Research Foundation – Flanders (FWO). This work was also supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20153. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

7. REFERENCES

[1] E. Agichtein and S. Cucerzan. Predicting accuracy of extracting information from unstructured text collections. In *CIKM*, 2005.

[2] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In *ICDE*, 2003.

[3] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[4] Z. Bar-Yossef and M. Gurevich. Efficient search engine measurements. *TWEB*, 5(4):18:1–18:48, 2011.

[5] P. Barrio, G. Simões, H. Galhardas, and L. Gravano. REEL: a relation extraction learning framework. In *JCDL*, 2014.

[6] P. Barrio, G. Simões, H. Galhardas, and L. Gravano. Learning to rank adaptively for scalable information extraction. In *EDBT*, 2015.

[7] M. K. Bergman. White paper: the deep web: surfacing hidden value. *JEP*, 7(1), 2001.

[8] C. Boden, A. Löser, C. Nagel, and S. Pieper. FactCrawl: A fact retrieval framework for full-text indices. In *WebDB*, 2011.

[9] R. C. Bunescu and R. J. Mooney. Subsequence kernels for relation extraction. In *NIPS*, 2005.

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

[11] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, 1995.

[12] Y. Fang and K. C.-C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *WSDM*, 2011.

[13] R. A. Fisher. *Statistical methods for research workers*. Number 5. Genesis Publishing Pvt Ltd, 1936.

[14] C. Giuliano, A. Lavelli, and L. Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL*, 2006.

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

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

[17] S. Gupta and K. K. Bhatia. A comparative study of hidden web crawlers. *IJCTT*, 12(3):111–118, 2014.

[18] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. *CACM*, 50(5):94–101, 2007.

[19] D. Hong, L. Si, P. Bracke, M. Witt, and T. Juchcinski. A joint probabilistic classification model for resource selection. In *SIGIR*, 2010.

[20] P. G. Ipeirotis and L. Gravano. Classification-aware hidden-web text database selection. *TOIS*, 26(2):6:1–6:66, 2008.

[21] A. Jain and D. Srivastava. Exploring a few good tuples from text databases. In *ICDE*, 2009.

[22] S. Krause, H. Li, H. Uszkoreit, and F. Xu. Large-scale learning of relation-extraction rules with distant supervision from the web. In *ISWC*, 2012.

[23] K.-L. Liu, A. Santoso, C. Yu, and W. Meng. Discovering the representative of a search engine. In *CIKM*, 2001.

[24] J. Madhavan, S. R. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, A. Halevy, and G. Inc. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, 2007.

[25] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *ICML*, 2000.

[26] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CONLL*, 2003.

[27] S. Raghavan and H. García-Molina. Crawling the hidden web. In *VLDB*, 2001.

[28] C. Sherman and G. Price. The invisible web: uncovering sources search engines can't see. *LIBT*, 52(2):282–298, 2003.

[29] M. Shokouhi and L. Si. Federated search. *FTIR*, 5(1):1–102, 2011.

[30] M. Shokouhi and J. Zobel. Federated text retrieval from uncooperative overlapped collections. In *SIGIR*, 2007.

[31] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR*, 2003.

[32] L. Si and J. Callan. Unified utility maximization framework for resource selection. In *CIKM*, 2004.

[33] M. van der Loo. Distribution based outlier detection for univariate data. Technical Report Discussion paper 10003, Statistics Netherlands, The Hague, 2010.

[34] M. Zhang, N. Zhang, and G. Das. Mining a search engine's corpus: Efficient yet unbiased sampling and aggregate estimation. In *SIGMOD*, 2011.

[35] M. Zhang, N. Zhang, and G. Das. Mining a search engine's corpus without a query pool. In *CIKM*, 2013.

[36] Z. Zheng, X. Wu, and R. Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explorations*, 6(1):80–89, 2004.