Towards Complete Coverage in Focused Web Harvesting

Mohammadreza Khelghati
Database Group
University of Twente, Netherlands
s.m.khelghati@utwente.nl

Djoerd Hiemstra
Database Group
University of Twente, Netherlands
d.hiemstra@utwente.nl

Maurice van Keulen
Database Group
University of Twente, Netherlands
m.vankeulen@utwente.nl

ABSTRACT

With the goal of harvesting all information about a given entity, in this paper, we try to harvest all matching documents for a given query submitted on a search engine. The objective is to retrieve all information about for instance "Michael Jackson", "Islamic State", or "FC Barcelona" from indexed data in search engines, or hidden data behind web forms, using a minimum number of queries. Policies of web search engines usually do not allow accessing all of the matching query search results for a given query. They limit the number of returned documents and the number of user requests. These limitations are also applied in deep web sources, for instance in social networks like Twitter. In this work, we propose a new approach which automatically collects information related to a given query from a search engine, given the search engine's limitations. The approach minimizes the number of queries that need to be sent by analysing the retrieved results and combining this analysed information with information from a large external corpus. The new approach outperforms existing approaches when tested on Google, measuring the total number of unique documents found per query.

Categories and Subject Descriptors

H.4 [**Information systems**]: Web searching and information discovery

Keywords

World Wide Web, Web mining, Data extraction, Deep web, Data coverage, Web harvester

1. INTRODUCTION

Nowadays, data is one of the keys to success. Whether you are a fraud detection officer in a tax office, a data(-driven) journalist, or a business analyst, your primary concern is to access all data relevant to your topic of interest. For a data journalist investigating a company, an in-depth analysis is infeasible without a comprehensive collection of data. This emphasizes the role of the web as one of the main gates to data. The availability of an up-to-date crawl of the web would definitely facilitate collecting all relevant information on a given entity. However, given the software and hardware requirements of crawling the web, this seems to be impracticable except for a few big organizations, and the journalist has to resort to using the search engines provided by such organizations.

Most web data is accessible by querying general search engines like Google, or Yahoo, or by submitting forms in deep web data sources. Achieving a full data coverage for an entity via either of these web data access methods has its own challenges. In [1, 15, 12, 5, 3], the focus is mainly on deep websites requiring form submissions. These studies investigate web forms, form fields' inputs, their bindings, and other features of the forms and websites which could influence harvesting a deep website and extracting information about a given entity. Instead, our paper examines search systems with keyword-based search interfaces[1]. This enables us to include any websites with keyword search.

With keyword search as the only data access approach, to achieve a full data coverage on a given query in a search system, the primitive solution is to follow these steps: 1) submit a query; 2) retrieve returned results in the results page; 3) go to next results page; and 4) Repeat number 2 and 3 till there is no next page. For most queries with thousands of results, going through these steps is a labour-intensive task. Web scrapers address this challenge by automatically navigating through search results and downloading desired data. However, even while using (semi-)automatic scrapers, achieving full data coverage on a given query in a search system is not as straightforward as it looks. Currently, search engines impose limitations that hamper retrieval of all returned results:

Limitation 1: *#ResultsLimited* The number of results a search engine allows a user to access is limited.

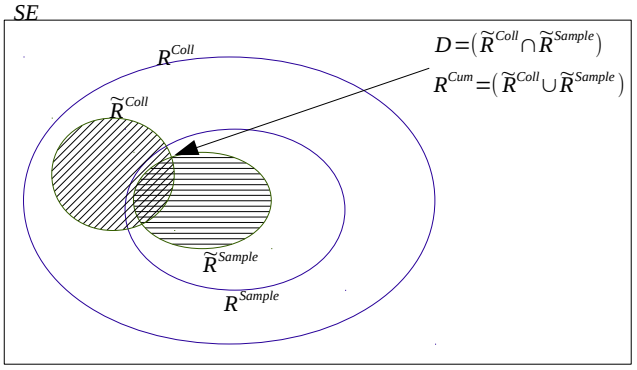
Limitation 2: *#RequestsLimited* The number of requests a user is allowed to perform within a certain period of time is limited.

[1]They provide a single text field to submit a query.

iiWAS '15, December 11-13, 2015, Brussels, Belgium

DOI: http://dx.doi.org/10.1145/2837185.2837208

Figure 1: Problem Definition



It is also worth mentioning that general search engines are designed to help users in finding answers to specific questions and not providing complete data coverage on a submitted query. For example, determining the location of a company's headquarters is not a challenging task with Google, or Yahoo search. However, retrieving all information about that company requires much effort if at all possible due to the above limitations.

This work relies on the potential power of search refinement techniques to uncover results beyond what a search engine allows a user to directly access due to #`ResultsLimited` and #`RequestsLimited` limitations. These techniques are typically based on adding extra terms to the initial query to obtain refined search results. We propose an approach which refines search results for the purpose of achieving full data coverage.

Problem Definition For a given query Q^{Coll}, R^{Coll} is the set of all documents containing the query. A search engine that imposes #`ResultsLimited` limitation, prevents users from accessing all these documents. With submitting Q^{Coll}, users cannot view more documents than a pre-determined number l. The l is the maximum allowed number but is not always the actual number of returned results. The returned results accessible for users for query Q^{Coll} is defined as $\tilde{R}^{Coll}$. Therefore, we have $|\tilde{R}^{Coll}| \leq l$. In case of overflowing queries[2], we have $l < |R^{Coll}|$ which leads to $\tilde{R}^{Coll} \subset R^{Coll}$. The documents returned for Q ($\tilde{R}^{Coll}$) cannot exceed the l. To retrieve all members of R^{Coll}, we form a new query Q^{Sample} by adding a term T to the Q^{Coll} query ($Q^{Sample} = Q^{Coll} + T$). The documents matching this newly formed query is referred to as R^{Sample} and the ones accessible by a user are defined by $\tilde{R}^{Sample}$. The documents present both in $\tilde{R}^{Sample}$ and $\tilde{R}^{Coll}$ are defined as duplicates and referred to as D. We also define R^{Cum} as the union of all retrieved documents. Provided that $\tilde{R}^{Sample}$ does not include only duplicates, it provides new documents from R^{Coll}. The Figure 1 models these definitions.

To obtain all documents in R^{Coll}, several iterations of query formulation could be performed. After each iteration of query reformulation and submission, the R^{Cum} and D are recalculated as shown in Formula 1. We continue query formulation and submission iterations till R^{Cum} equals R^{Coll}. Now, the *main problem* to be addressed is how to minimize the number of these iterations to have an efficient approach which complies also with the #`RequestsLimited` limitation.

$$R_n^{Cum} = \bigcup_{i \in 1,n} \tilde{R}_i^{Sample}$$
$$D_i = \tilde{R}_i^{Sample} \cap R_{i-1}^{Cum} \quad (1)$$

In this approach, reformulating queries should be carried out with the aim of obtaining as many new results as possible for each query. Maximizing the number of new results means submitting queries which return as many documents as l while minimizing the number of D. This defines the goal of this work as follows:

Our goal: Collecting the biggest possible set of documents in a search system that match a given query by posing the least possible number of requests.

Therefore, maximizing $\tilde{R}^{Coll} = l$ while minimizing D are necessary requirements to achieve this goal. Minimizing duplicates becomes complicated with the presence of *ranking bias* and *query bias* [4]. Search engine's ranking algorithms (e.g. Google Page-rank) and selection of the initial query favour some documents more than others to be returned by the search engine.

Therefore, we define the *main challenge* in this paper as finding counter-measures for biases generated by ranking algorithms. These biases cause same documents to be ranked always high among returned results and therefore always present in $\tilde{R}^{Coll}$ and $\tilde{R}^{Sample}$s which leads to a big D. To meet this challenge, several approaches are suggested, implemented, and compared in this paper. We test our approaches on Google, which claims to search 100 PB of Web data (60 trillion URLs)[3]. Google imposes both #`ResultsLimited` and #`RequestsLimited`, and ranking bias through its Page-Rank algorithm.

This work is a follow-up on our previous work published as a short paper [14]. In that short paper, we first introduced the problems in harvesting all the matching documents for a given query and suggested a number of approaches based on the information extracted only from an external corpus. In this work, we introduce a formal problem definition to further clarify the issue. In addition, In addition, the new concepts of `Completeness` and `Relevance` are introduced to give a better understanding of the topic. Also, instead of relying on external corpora, the returned results for query submissions are used as feedback mechanisms.

The next section is dedicated to related work. Then, Section 3 discusses solutions classified into three main categories. In Section 4, these solutions are tested and the results are analysed and presented. Finally, in Section 5, conclusions drawn from these results are discussed and areas for further research are identified.

2. LITERATURE STUDY

Deep Web harvesting In this work, we are interested in the methods that are applied to access deep web data either for sampling or harvesting websites. Recent studies on accessing deep web data focus mainly on websites requiring form submissions [1, 15, 12, 5, 3], studying the website form to achieve an efficient data access approach. For example,

[2]An overflowing query is a query that produces more results than #`ResultsLimited` allows.

[3]Official Google Blog: http://googleblog.blogspot.nl/2008/07/we-knew-web-was-big.html

in [15], authors investigate how to identify the values and types of different form fields, and how to efficiently navigate the search space of possible form input combinations to avoid unnecessary submissions leading to a more efficient source sampling process. Although the goal of these studies is not entity focused harvesting, the idea of devising different query generation plans in querying a data source is relevant. These studies focus on the features of forms, and require additional form analysis and extra knowledge.

Query-Based Sampling Accessing data in all diverse web sources with search interfaces follows a similar method, referred as query based sampling (QBS): submitting requests to get results pages. In QBS, by sending a query to the search engine, the set of returned results is considered as a sample of documents [6, 4]. Having produced a number of these samples, they are used in different statistical calculations (e.g. search engine size estimation, or quality test) where it is important to have randomly generated samples. However, in search engines like Google, the application of complicated ranking mechanisms violates this randomness assumption and accordingly generates uncertainties in the calculations. In sampling documents, factors such as chosen query, content of documents, ranking mechanism and many others affect the probability of a document to be selected and therefore the samples' randomness. Non-random samples make calculations based on QBS doubtful. To solve this, in [2, 13, 19], a number of different approaches are introduced. However, none of these approaches aim at reducing duplicates in samples but on keeping calculations free from the biases. Oppositely, in this work, we require an approach to remove biases effects in the generated samples resulting in a smaller D among them. Generating random or close-to-random samples can be considered as one of the counter-measures against generated biases.

Topical Crawling In *focused crawlers* which are also known as *topical crawlers*, a crawl starts from a user-provided set of data and collects results only for his given topics [17, 20]. The focus of these crawlers is to locate relevant pages without crawling all links. To guide this navigation, with the usage of available contextual information (e.g. links, and content of previously crawled pages), different techniques such as link analysis, automatic classification, text analysis, machine learning, and evolutionary algorithms are applicable [20] helping to estimate the relevance of a page to a given topic. These crawlers are designed for particular information needs expressed by topical queries or interest profiles [17].

Query Expansion Query expansion is the process of reformulating the original query with the goal of getting a better query that improves retrieval effectiveness, a query that is more likely to retrieve relevant documents leading to better top-k recall and precision [8]. Automatic query expansion techniques can be classified into five main groups according to the techniques used for finding the expansion features: linguistic methods, corpus-specific statistical approaches, query-specific statistical approaches, search log analysis, and web data [8]. The pseudo-relevance feedback from query-specific statistical approaches category is one of the more interesting approaches for the purpose of this paper where it is assumed that terms in retrieved documents are useful for the retrieval task and expanding the original query. The criteria for selecting among these terms can be based on a number of different features of the documents and terms [7, 16, 8, 9, 11].

3. ENTITY-FOCUSED WEB HARVESTING

Before diving into solutions, we need to clarify two concepts; completeness, and relevance.

Completeness We formulated our goal as collecting "all information" available on the web for a given entity. Motivated by this goal, we consider the first step to gather all the pages including that information. As we will show in 4, this is still a challenging task. Having gathered all the relevant pages, the next step is to extract information. However, this is beyond the scope of this paper and is suggested as future work. Therefore, this paper focuses on finding the most efficient approaches to get the biggest possible coverage in terms of documents for a given entity.

Relevance Looking for an entity, we gather all the pages that contain that entity. These pages are downloaded and indexed. These pages are used to extract the entities. However, the focus is only on pages which include the same terms as the given entity and not the extracted entities.

To reach this data coverage, we send automatically generated queries to a search engine's API with the goal of retrieving all documents that contain a given entity with a minimum amount of query submissions. We compare the approaches by their capabilities to deal with #`ResultsLimited` and #`RequestsLimited`. The comparison is based on the average number of queries submitted to retrieve all documents for a given query. We distinguish three kinds of approaches. Section 3.1 describes ideal approaches, for which we estimate the number of queries needed in ideal (simulated) conditions. Section 3.2 describes approaches in which queries are reformulated by using an external corpus. Section 3.3 focuses on approaches inspired by the pseudo-feedback methods for query expansion. In this method, extracted content from the previously retrieved documents is the source for the query generation process.

3.1 Ideal Approaches

The approaches mentioned in this section are desirable or perfect but not easily realized. These are investigated with the sole purpose of improving the comparison of the introduced approaches. The Oracle Perfect approach [14] never retrieves a duplicate document. In this paper, we consider a more realistic goal by using a probability of retrieving a duplicate, which is more informative in comparison to the other approaches. This is discussed in more details in the Results Section.

3.1.1 Oracle Perfect Approach

To download all the matching documents for a given query from a search engine which imposes the #`ResultsLimited` and #`RequestsLimited` limitations, a perfect approach returns not only the maximum possible number of documents (l) but also only unique ones for each request submitting either the Q^{Coll} or the Q^{Sample}s. Submitting only the $\frac{|R^{Coll}|}{l}$ number of requests will cover all matching documents [14]. In reality, this is not easily reachable cause of lack of information on ranking algorithms and term frequencies. With this information on ranking algorithms and term frequencies, one can divide the collection into exactly $\frac{|R^{Coll}|}{l}$ subcollections. This is only accessible when you have full index access which does not happen in reality.

3.1.2 Probability Based Approach

Provided that there is a uniform selection probability for all documents matching a given query in a search system, we have no bias in selecting documents to return as query results. This is only the case in search engines without query and ranking biases. These biases are direct results of ranking algorithms and the criteria of documents matching a given query to be returned as results. In a system in which all query results are drawn uniformly at random, we can generate random samples. With this assumption, statistical formulas can be applied to calculate the predicted number of duplicates and accordingly the number of unique results in a set of randomly generated samples. The following formula calculates the estimated number of unique documents and duplicates in any of the query submissions to a search engine.

$$|R_n^{Cum}| = |R^{Coll}| - |R^{Coll}| * (1 - \frac{l}{|R^{Coll}|})^n \quad (2)$$

Formula definition Formula 2 calculates the number of newly discovered documents through all submitted queries to a search engine. It is assumed that the documents have a uniform selection probability. In this case, this probability is defined as $\frac{l}{|R^{Coll}|}$. Consequently, the probability of a document not to be selected is defined as $(1 - \frac{l}{|R^{Coll}|})$. Now, the goal is to calculate these probabilities after n number of query submissions (sampling events). Keeping this in mind that having a unique document in the n^{th} sampling event requires that document not to be selected in the $(n-1)$ previously generated samples. With $(1 - \frac{l}{|R^{Coll}|})$ as the probability of a document not to be selected in one sampling event, through $(1 - \frac{l}{|R^{Coll}|})^n$ the probability for a document not being selected in n sampling events can be determined. Multiplying this probability in the total number of documents matching the given query Q (R^{Coll} determines the number of not retrieved documents by the n number of previous query submissions. Subtracting this number from $|R^{Coll}|$ gives the number of all the previously retrieved documents. The formula is verified by comparing its output with the results of a simulation in which random samples were generated to measure the number of duplicates and unique documents.

3.2 List-Based Query Generation Approaches

In List-Based Approaches [14], the terms to be added to the seed query are selected from a list of words generated from an external corpus (ClueWeb09 dataset). In [14], three different List-Based approach are suggested based on selecting terms either with most, least or a pre-determined frequency to reformulate the original query. In [14], it is shown that the pre-determined frequency approach (*LB-FixedFreq.*) outperforms the other approaches. This approach is included in our experiments as the best performing approach of [14].

3.2.1 Pre-determined Frequency Based Approach

With the most and least frequent terms increasing the chance of maximum returned results and fewer duplicates respectively, this approach investigates a term frequency resulting in a trade-off between these two extreme cases. Considering each query submission as an independent event, the probability of having an overlap between two queries equals with the multiplication of the probability of each query ($P(A\&B) = P(A) * P(B)$). This is shown in Formula 3.

$$\frac{|R^{Coll} \cap R^{Sample}|}{s^{SE}} = \frac{|R^{Coll}|}{s^{SE}} * \frac{|R^{Sample}|}{s^{SE}}$$

$$|R^{Sample}| = \frac{l * s^{SE}}{|R^{Coll}|} \mid (|R^{Coll} \cap R^{Sample}| = l) \quad (3)$$

From this formula, with information on the number of documents matching the seed query, returned results and search engine size, a term can be found to formulate a new query returning at least l results. To get information on terms documents frequencies, full access to search engine index is necessary. As this is not possible in most of web search engines, pre-computed terms documents frequencies from an external corpus (such as ClueWeb [18]) can be used [13]. If the size of search engine is unknown, as discussed in [13], the size can be estimated by only using a few number of generated samples from search engine.

For example, assuming $s^{SE} = 10^9$, the number of English documents in ClueWeb as 5×10^8, $l = 100$, and $|R^{Coll}|$ for a given query to be 4×10^5, the following calculation could provide us with a term document frequency that has higher chance to result in samples of our desired size: $\frac{100}{10^9} = \frac{4\times10^5}{10^9} * \frac{x}{5\times10^8} \Longrightarrow x = 125000$. This approach is referred to as *LB-FixedFreq.*.

3.3 Feedback-Based Approaches

In this section, terms to reformulate queries are selected from the previously retrieved content. The criteria for this selection can be based on a number of different features of terms and documents.

3.3.1 FB-Most Frequent Terms Based Approach

Among the terms extracted from downloaded documents, the ones with higher frequencies have a higher chance in returning at least l results. This is the same principle as introduced in Subsection **??**. This approach commences by submitting a query to the search engine. Then, having extracted the content of the results for the query, the most frequent word in that content is selected to be used in query reformulation. The final step is adding this most frequent word to the original query and submitting the constructed query to the search engine. With new results obtained from this query submission, these steps are repeated and new queries are formed and submitted. This process repeats till reaching a full data coverage for the original query. For example, with the goal of reaching full data coverage on the term "Vitol" which is an oil company, the term "Oil" appears as the most frequent term in the returned documents by search engine. The next step is submitting "Vitol"+"Oil". This approach is referred to as *FB-MostFreq.* approach.

3.3.2 FB-Least Frequent Terms Based Approach

In this approach, against the FB-Most Frequent Terms Based Approach, the least frequent terms are selected. The reasoning behind this difference is the struggle for minimizing the number of duplicates among the generated samples. For example, in the first step for covering all related documents for the term "Vitol", the term "damit" which is a German word is selected to be used in reformulating the query. This approach is referred to as *FB-LeastFreq.* approach.

3.3.3 FB-Least from Last Approach

In this approach, to counteract the effects of ranking algorithms of search engines in favouring a number of documents more than others, the terms for query reformulations are selected from the pages with lower ranks. This means the terms are selected from the documents present in the bottom of the returned results for a query. This is due to search engine's behaviour in returning more important and relevant pages always on top. Targeting the results in the bottom of the results list is for selecting terms which are negatively correlated with the original query and therefore, have higher chance in generating fewer duplicates among the samples. Therefore, in this approach, the least frequent word in the the last returned search result is selected to be used for query reformulation in next steps. This approach is referred to as *FB-LeastFromLast* approach.

3.3.4 FB-Fixed Frequency Based Approach

In Subsections 3.3.2 and 3.3.1, the most and least frequent terms in the retrieved documents are selected to reformulate queries. These frequencies represent extreme cases. The most frequent words represent higher chances in returning the most allowed number of queries and the least frequent terms produce potentially less duplicates. A balanced approach which includes both these cases is also in the interest of our investigation. In defining such an approach, the formula introduced in 3.2.1 subsection is applicable. This formula calculates a specific frequency that has higher chance to result in samples of our desired size (l). However, this formula is applicable only in selecting terms from an external corpus with a pre-calculated list of terms and frequencies. To apply this formula to a feedback-based approach, it is required to determine the corresponding frequency in the retrieved documents with the resulted frequency from the mentioned formula. To do so, this formula is applicable:

$$T.D.F.^{FeedbackText} = \frac{|R^{Sample}| * |R^{Cum}|}{s^{SE}} \quad (4)$$

In this formula, the $|R^{Sample}|$ is resulted from Formula 3 defined in 3.2.1. With the number of retrieved documents ($|R^{Cum}|$) and the size of search engine (s^{SE}) known, the document frequency for the next term selection is determined ($T.D.F.^{FeedbackText}$). It is assumed that the terms with this frequency in the retrieved documents have higher chance to result in samples of the desired size with fewer duplicates as this approach targets making a trade-off between the extreme cases of submitting the most and least frequent terms. This approach is referred to as *FB-FixedFreq* approach.

3.4 Combined List-Feedback Based Approach

In the list-based approaches, the terms are selected from an external corpus with a specific pre-determined frequency. However, this frequency as a selection criteria does not offer any information about the potential relevance between terms and original queries. To include this missing information in the selection process, terms which are present both in the list of external corpus terms and in previously retrieved documents are selected. For example, as the next query after submitting "Vitol", we choose a term appearing in both the retrieved content and the list from the ClueWeb dataset. This helps to have both relevance and frequency effects in one approach. This approach is referred to as the *Comb.LB-FB* approach.

4. EXPERIMENTS AND RESULTS

4.1 Experiments Settings

Test Search Engine In this paper, we use Google as the biggest web search engine with one of the most complicated ranking algorithms as our test search engine. As support of keyword-search interface is the only prerequisite to apply any of the suggested approaches, targeting Google does not limit our findings. If the suggested approaches work for Google, they have a good chance in reaching good performance on any other website too.

Although Google is used only as an example of a search engine in our paper, we find it necessary to provide evidence to support the discussed prerequisites even for this example. In Google, a user can submit at most 100 queries a day [10]. Through our experiments, we noticed no more than 500 results are accessible for any submitted query. This number was dynamic but in all our experiments, it was less than 500. With Google Web Search API being depreciated, Google Custom Search, is actually capable of searching the entire web, but it suffers from the same limitations. As an alternative, Google Site Search eliminates #ResultsLimited at the expense of being able to search the entire web. Google Site Search permits you to search only a specific set of web sites. In consequence, your results are unlikely to match those returned by Google Web Search and also different from 'site:' search on Google.com. There are also costs for submitting more than 100 search queries per day.

Entities Test Set In our experiments, 120,000 queries were submitted to download information for four different entities ("Vitol", "Ed Brinksma", "PhD Comics", and "Fireworks Disaster"). These entities represent diverse types of entities; Company, Person, Topic, and Event. In addition to difference in type, we tried to cover queries with different estimated results sets sizes ranging from 2×10^4 to 5×10^5.

Relevance Judgement In assessing an approach in achieving full data coverage on an entity, the amount of retrieved data is the decisive metric. However, it needs to be further clarified what is considered as retrieved data. In our experiments, we consider the number of documents returned by search engine for an entity (R^{Cum}) as retrieved data. Therefore, all the retrieved documents that contain the searched keyword are considered as related. There are other possible options which are mentioned as future work. As all the documents contain the keywords, the more documents retrieved increases the chance to reach more completeness.

Evaluation Metric Assessing different approaches for one entity is straightforward by comparing the R^{Cum}s of all the approaches. However, the general performance evaluation of these approaches on all the entities is not possible only through comparing R^{Cum}s. With different results set sizes for entities, comparing $R^{Cum}_{n_{Entity1}}$ with $R^{Cum}_{n_{Entity2}}$ of the same approach does not reveal much information about its overall performance. While for a small results set, a few number of queries can cover all documents, for a bigger collection, that number of queries can just retrieve a very small part. Therefore, we evaluate each approach for a given entity by its distance from performance of the Probability-based approach for that entity (Section 3.1). This is calculated through Formula 5. These distances are calculated for all the other entities in the entities test set and averaged to represent the

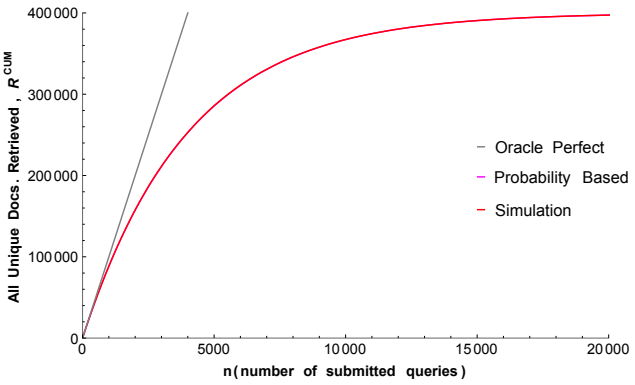


Figure 2: Retrieved Documents (R^{Cum}) for Ideal approaches

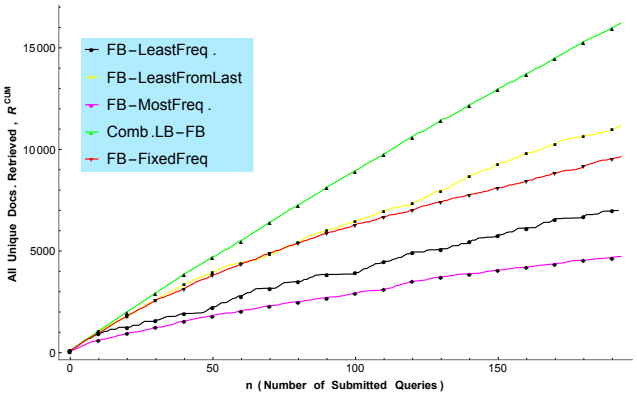


Figure 3: Retrieved documents (R^{Cum}) for FB-Based approaches on one entity

general approach performance.

$$Performance = \frac{|R^{Cum}_{n_{Prob.Appr}}| - |R^{Cum}_{n_{Test.Appr}}|}{|R^{Cum}_{n_{Prob.Appr}}|} \times 100 \quad (5)$$

Fixed l One of the main challenges for data coverage is the limitation on the number of returned results. In Google, this number is not fixed. In our experiments, this number changed from 200 to 500 even for the same query but at different times. It seems that Google acts randomly (or based on a set of reasons which are not known to us). This creates an uncertainty on the size of samples for our experiments. In all the experiments in this paper, the sample size is set to 100 to assist comparisons and increase reliability in conclusions. *Practical Details* There are also a number of small practical decisions like what to choose as the first query or the usage of quotation marks in the query (phrase queries) which should be noticed. In this work, we always submit queries between quotation marks.

4.2 Results

In this section, the results of applying the introduced approaches in Section 3 to the test entities (Section 4.1) are presented. To establish a comparison baseline, in Figure 4.2, the Oracle Perfect, Probability, and simulation-based approaches (3.1.2) are compared in retrieving a collection of 4×10^5 documents with $l = 100$. The Oracle Perfect outperforms the other approaches with all samples of the maximum size and no duplicates. The Probability-based approach performs worse than the Oracle Perfect but the same as the simulation-based. In both Probability-based and simulation-based approaches, the selection of documents is random.

The results of running the FB-Based approaches (Section 3.3) are shown in Figure 4.2. This figure plots the performances of the approaches in obtaining unique documents after each query submission. For an x-axis value, the approach with the highest corresponding y-axis value returns more unique documents. As observable from this figure, among the FB-Based approaches, the Comb.LB-FB approach outperforms the others.

To better understand the reasons behind this good performance, in Figures 4 and 5, the potential influences of sample size and number of duplicates are studied. In Figure 4, there are four scatter plots for four of the introduced approaches. In each of these scatter plots, each point shows the relationship between the sample size and the number of duplicates for each query submission. The sample size is plotted along the x axis, and the y axis gives the number of returned results. The pattern of the resulting points of plotting all the values for all the query submissions reveals the correlation between these two variables and the approach performance. This pattern gives an overall view on the samples sizes and their numbers of unique documents. In Figure 4, in each of the four graphs, the points in the top right corner represent samples of maximum size and fewer duplicates. To show how sample sizes and number of duplicates differ among the approaches, in Figure 5, the samples sizes and duplicates of all the FB-based approaches are shown separately in two different graphs. In Figure 5, the right graph shows the number of duplicates for each query submission and the left graph presents the samples sizes of the submitted queries. In the left graph, a dot placed higher than others present a sample with more number of results than other dots and hence it is more desirable. However, the points in the lower part of the right graph in Figure 5 are more desirable as they represent the fewer number of duplicates.

Figure 4.2 compares the performances of the approaches only for one entity, whereas Figure 4.2 compares the average performances of all the approaches for all the entities in the test set. These average performances are calculated through the Formula 5 in Subsection 4.1. For each entity, for all the iterations of query submissions, the performance of the Probability-based approach is calculated. For all the introduced solution approaches, their distances from this Probability-based approach are calculated. For all the other entities, these calculations are repeated. At the end, the average distances are calculated for each approach over all the entities. Using the Probability-based approach as a baseline enables a better comparison as the size of each entity is already included in the calculations. As it is observable from Figure 4.2, there are big gaps between each approach and the estimated probability. This emphasizes the effects of ranking algorithms of search engines.

4.3 Analysis

As illustrated in Figures 4 and 5, the key to success (having more data coverage) is bigger samples with fewer duplicates. However, there is a trade-off between these two goals.

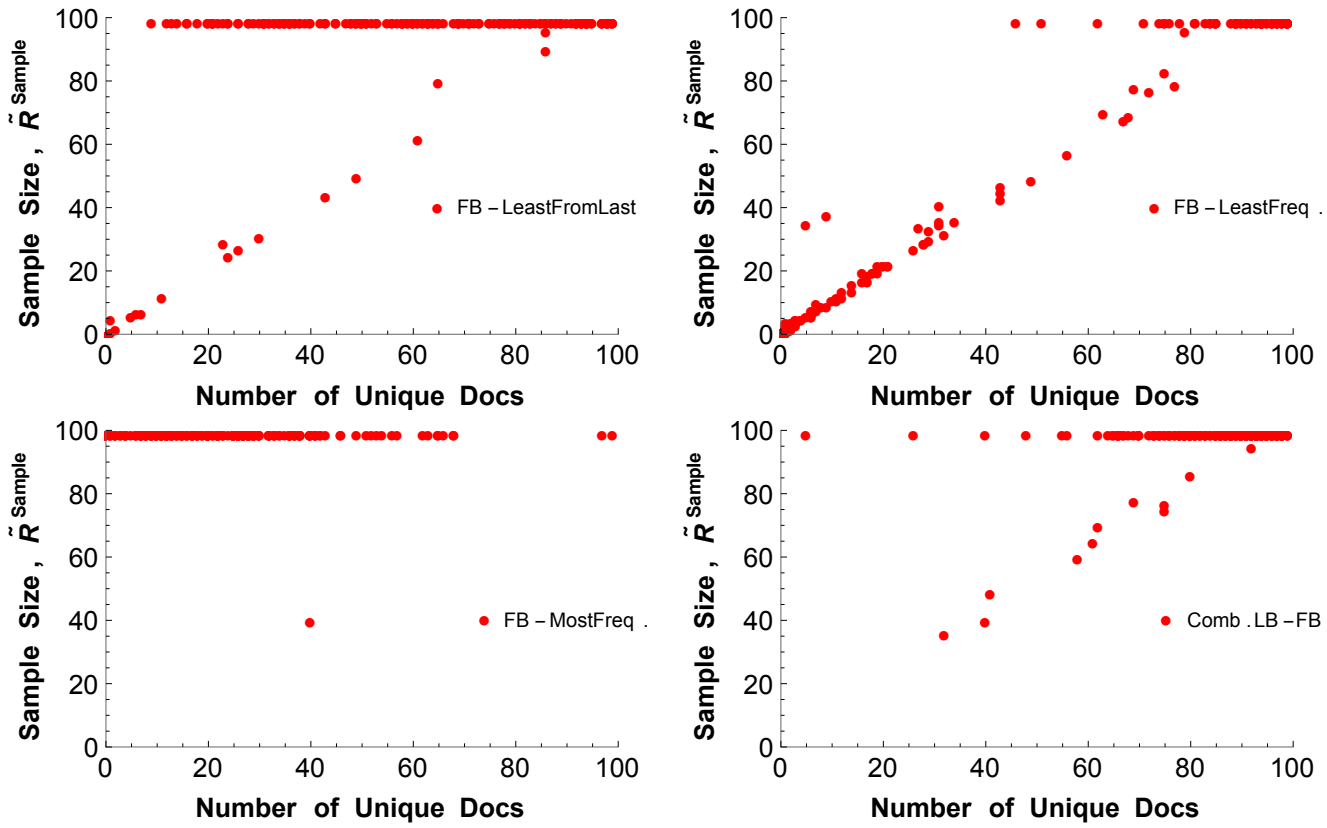


Figure 4: Sample size and unique Docs of generated samples for FB-Based Approaches

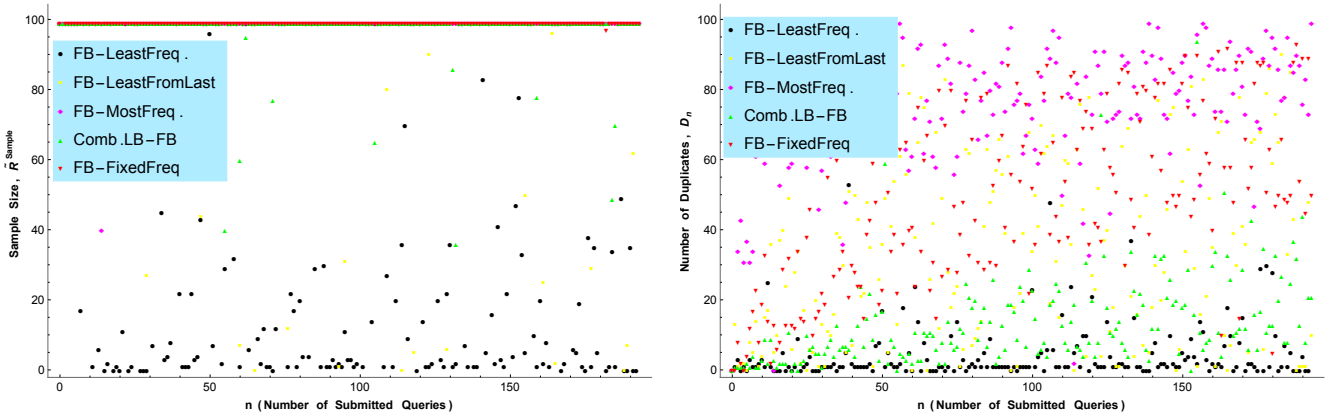


Figure 5: Sample size and duplicates of generated samples for FB-Based Approaches

Bigger samples increase the chance of more duplicates. Instead, to reach fewer duplicates, smaller samples are helpful. In the FB-MostFreq. approach (Section 3.3.1), samples are big but but include a lot of duplicates. In the FB-LeastFreq. (Section 3.3.2), this is exactly the other way around.

One way to tackle the challenge of achieving this trade-off is to find a specific word frequency to submit as next query. In the LB-FixedFreq. approach (Section 3.4), different frequencies are applied. In our experiments, we observe that the low and high frequencies yield worse coverage and the highest coverage was the result of submitting words with a frequency derived from Formula 3.

Another way to achieve this trade-off is to counteract the biases of search systems in selecting documents like PageRank-generated bias in Google. The introduced FB-ListFromLast approach, which is based on this bias removal idea, produced better results than FB-MostFreq and FB-LeastFreq. However, this approach could not outperform the Comb.LB-FB. approach. In Figures 4 and 5, it is observable that the number of duplicates and sample sizes for the Comb.LB-FB. approach are more desirable than the ones in the FB-ListFromLast approach.

Among all the introduced approaches, the Comb.List-FB approach performs the best. In this approach, the queries

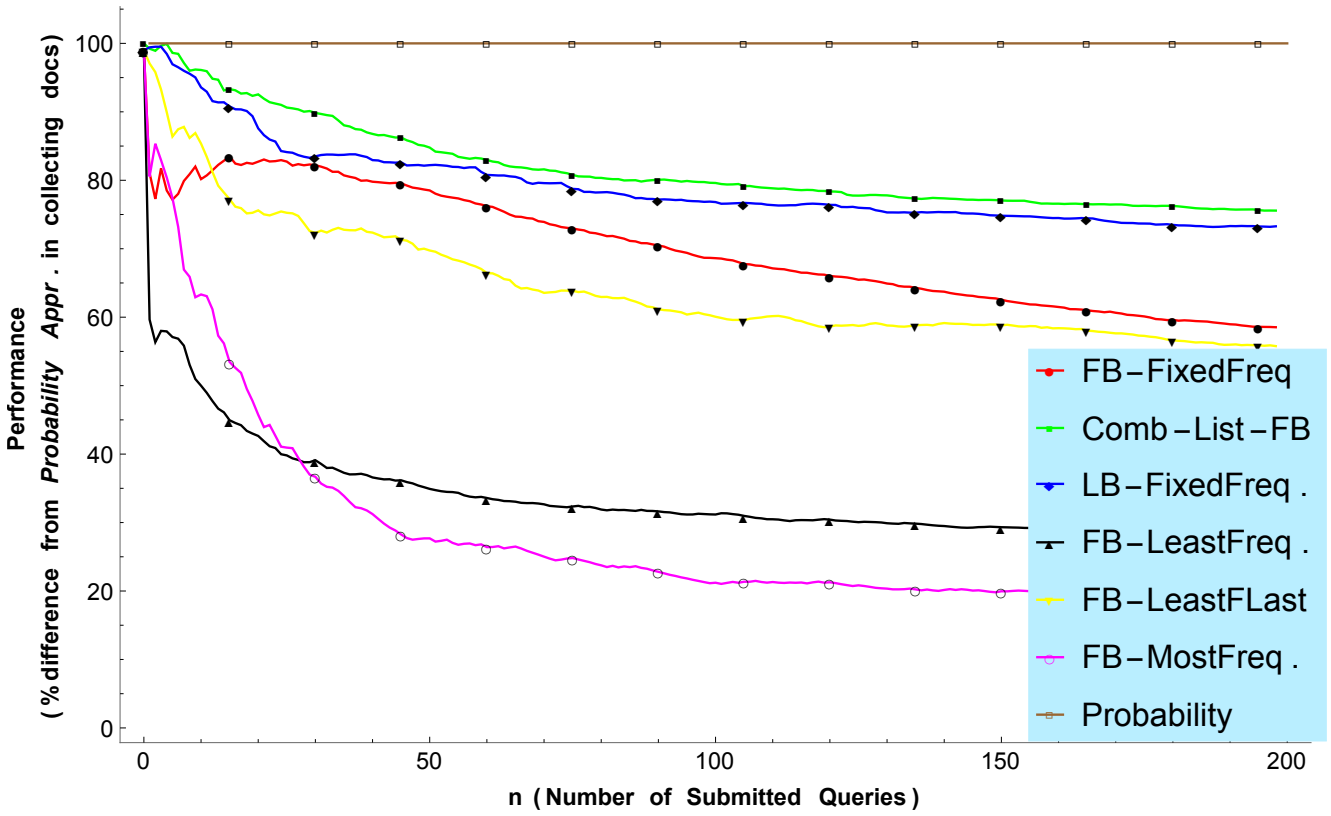


Figure 6: Average Performance For all Queries

submitted in specific frequency are refined to reduce the number of small samples. This causes 10 percent better documents coverage in average for all the submitted entities.

5. CONCLUSION AND FUTURE WORK

In this work, we assessed different query generation mechanisms for harvesting a web data source to reach a full data coverage on a given entity. From the experiments, we found that the key to success in these approaches is to send queries which result in the maximum possible number of results with the minimum possible number of documents downloaded in previous query submissions. To have this success factor, we suggested different approaches based on different frequencies and possible dependencies. From these approaches, the Comb.List-FB approach which analysed the retrieved results and combined this analysed information with information from a large external corpus performed the best.

Future Work In addition to word frequency and presence of word in retrieved documents for selecting the best next query to submit, there are a number of issues such as terms distribution in returned documents, their distances, and dependency of candidates to all previously submitted queries which might help in counteracting the search systems biases. The future work can focus on experimenting the effects of these factors. As another next step, the retrieved documents can be disambiguated, analysed, and the entities in those documents can be extracted. For example, searching Vitol results in different topics from a company, to a person. Analysing the returned results by different techniques like documents clustering helps refining retrieved documents and analysing only the ones of user interest.

6. ACKNOWLEDGEMENT

This publication was supported by the Dutch national program COMMIT.

7. REFERENCES

[1] Manuel Álvarez, Juan Raposo, Alberto Pan, Fidel Cacheda, Fernando Bellas, and Víctor Carneiro. Deepbot: a focused crawler for accessing hidden web content. In *Proceedings of the 3rd international workshop on Data enginering issues in E-commerce and services: In conjunction with ACM Conference on Electronic Commerce (EC '07)*, DEECS '07, pages 18–25, New York, NY, USA, 2007. ACM.

[2] Ziv Bar-Yossef and Maxim Gurevich. Efficient search engine measurements. *Proceedings of the 16th international conference on World Wide Web*, pages 401–410, 2007.

[3] Luciano Barbosa and Juliana Freire. Siphoning hidden-web data through keyword-based interfaces. In *SBBD*, pages 309–321, 2004.

[4] Krishna Bharat and Andrei Broder. A technique for measuring the relative size and overlap of public web search engines. *Comput. Netw. ISDN Syst.*, 30:379–388, April 1998.

[5] Michael Cafarella. Extracting and Querying a Comprehensive Web Database. In *Proceedings of the*

Conference on Innovative Data Systems Research (CIDR), 2009.

[6] James P. Callan and Margaret E. Connell. Query-based sampling of text databases. *ACM Trans. Inf. Syst.*, 19(2):97–130, 2001.

[7] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 243–250, New York, NY, USA, 2008. ACM.

[8] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50, January 2012.

[9] Kevyn Collins-Thompson and Jamie Callan. Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 303–310, New York, NY, USA, 2007. ACM.

[10] Google. Google custom search. https://developers.google.com/custom-search/, 2015.

[11] Ben He and Iadh Ounis. Combining fields for query expansion and adaptive query expansion. *Inf. Process. Manage.*, 43(5):1294–1307, September 2007.

[12] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 355–364, New York, NY, USA, 2013. ACM.

[13] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Size estimation of non-cooperative data collections. IIWAS '12, pages 239–246, New York, NY, USA, 2012. ACM.

[14] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Harvesting all matching information to a given query from a deep website. In *1st International Workshop on Knowledge Discovery on the Web (KDWEB'15)*, CEUR Workshop Proceedings, Aachen, 2015. (in press).

[15] Jayant Madhavan, David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy. Google's Deep Web crawl. *Proc. VLDB Endow.*, 1(2):1241–1252, August 2008.

[16] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[17] Filippo Menczer, Gautam Pant, and Padmini Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology*, 4:http://dollar.biz.ui, 2004.

[18] The Lemur Project. A dataset to support research on information retrieval and related human language technologies. http://lemurproject.org/clueweb09.php, 2014.

[19] Milad Shokouhi, Justin Zobel, Falk Scholer, and Seyed M. M. Tahaghoghi. Capturing collection size for distributed non-cooperative retrieval. In *SIGIR*, pages 316–323, 2006.

[20] Sergej Sizov, Martin Theobald, Stefan Siersdorfer, Gerhard Weikum, Jens Graupmann, Michael Biwer, and Patrick Zimmer. The bingo! system for information portal generation and expert web search. In *CIDR*, 2003.