Efficient Web Harvesting Strategies for Monitoring Deep Web Content

Mohammad 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

1. ABSTRACT

The change of the web content is rapid [23, 22, 23]. In *Focused Web Harvesting* [**?**], which aims at achieving a complete harvest for a given topic, this dynamic nature of the web creates problems for users who need to access a complete set of related web data to their interesting topics. Whether you are a fan following your favourite artist, athlete or politician, or a journalist investigating a topic, you need to access all the information relevant to your topics of interest and keep it up-to-date over time. General search engines like Google apply different techniques to enhance the freshness of their crawled data. However, in Focused Web Harvesting, we lack an efficient approach that detects changes of the content for a given topic over time. In this paper, we focus on techniques that allow us to keep the content relevant to a given entity up-to-date. To do so, we introduce approaches to efficiently harvest all the new and changed documents matching a given entity by querying a web search engine. One of our proposed approaches outperform the baseline and other approaches in finding the changed content on the web for a given entity with at least an average of 20 percent better performance.

2. INTRODUCTION

The change of the web content is rapid [23, 22, 23, 8, 16, 5, 7, 30, 8, 6, 10, 17]. This dynamic nature of the web data creates difficulties for users who need to access a complete collection of the web data for their queries. As a fan following your favourite artist, athlete, or politician or a business analyst studying stock market, you need to access all and the latest information that is relevant to your topic of interest. You need to have all the relevant information and you need to keep them up-to-date over time. With an crawl of the whole web that is kept up-to-date over time, the mentioned user needs can be satisfied. However, this seems impracticable even for big organizations and governments. Instead, as most of the web data is accessible by querying different search engines, the users have to resort to using these available search engines and collect all the relevant information for their queries of interest. This collection of data can be gathered, to some extent automatically, by applying focused web harvesting techniques introduced in [**?**]. In focused web harvesting, all documents matching a given entity are harvested by querying a web search engine. For instance, all information about "Bernie Sanders", "Islamic State", or âĂIJGolden Ball AwardâĂİ are retrieved from indexed data in search engines, or hidden data behind web forms.

In focused web harvesting, search engines usually limit access to all the matching results for a given query, number of returned documents and also user requests. These imposed limitations affect having an up-to-date collection. To have an up-to-date data collection related to a given topic, we need approaches that can efficiently return the documents with changed content, in addition to removed or newly created ones. Our ultimate goal is to find methods that harvest only the changed data instead of running another round of harvesting. Considering the costly process of harvesting, it is important to find methods that facilitate efficient reharvesting processes with the goal of retrieving only changed documents.

Given the ever changing nature of the web [23, 22, 23], keeping an up-to-date collection of data is a challenging task even for search engines with huge amount of available resources. For example, Google needs to keep billions of URLs[1] up-to-date. They apply different techniques to enhance the freshness of their indexes. We study the techniques in crawlers and big search engines that are aimed at increasing the freshness of their indexes. We also explore the page change detection algorithms in these systems. However, as in focused web harvesting targets only a small part of the web that is related to a topic, exploring the parts of these techniques resolving the size issues are not included.

Having explored web crawlers, their techniques in keeping an index up-to-date, change detection in crawlers, identical and near identical pages detection, we try to solutions for efficient retrieval of changed content on the web in the domain of focused web harvesting.

Contributions. As our first contribution, we study the change rate in the FedWeb data sets [3, 15] from two different years. In this study, we analyse the change rates of 150 websites

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

and 24 different categories.

As the second contribution, we study 4 different methods with the goal of finding the most efficient approach for retrieving the changed documents on the web. We test these approaches on our test search engine and report the results. The reports show that we can improve the retrieved changed content by at least 20 percent.

Sections. In the next section, Section 3, we study the methods applied by general search engines to keep their indexes up-to-date. We also discuss the freshness concept and how it is related. The techniques applied in the literature of web crawling to detect changes in pages are also studied. In addition to discussing the page change detection techniques for web crawling, we study the other available approaches to detect identical or near-identical HTML pages in Section 4. Section 5 discusses the change rate of websites and their categories. In Section 6, the suggested approaches to have efficient re-harvesting are described. The results of testing these approaches on our test set is presented in Section 7 and analyzed in Section 8. Section 9 draws conclusions and further future work.

3. LITERATURE STUDY

As mentioned in Introduction Section, the focused web harvesting domain shares the same concerns as web crawlers and search engines domain as they both need to update their local view of the web to reflect its changes. From web crawlers and search engines domain we are interested to explore the applied strategies and techniques to discover new pages, keep their indexes up-to-date and detect similar pages. While studying these strategies and techniques, we should have in mind the basic difference between focused web harvester and a crawler that is the role of queries in our focused harvesting against the use of links and URLs in crawlers.

3.1 Web Crawling and Focused Web Harvesting

Search engines use web crawlers to collect pages from the web [7]. In general, a web crawler starts with the initial URL known as seed URL [27]. Web pages corresponding to each URL are fetched (if robots.txt file allows) and parsed to extract hyper-links. The web pages of these hyper-links go through the same process of parsing and extracting. These steps are repeated till all extracted URLs are visited.

As the web is a huge collection of documents, by the time a web crawler has finished its crawl, events like creations of new web pages, page content updates, and pages deletions have already made big changes in the web content [4]. According to an estimation in [12, 27], the surface web currently contains more than 4.16 billion web documents which shows an enormous growth in comparison to the estimated 800 million pages in 1999 by Lawrence et al. in [22].

In addition to new pages, the web has a very dynamic nature and high rate of change. The frequency of the web document change has been studied in previous work [8, 16, 5, 7, 30, 8, 6, 10, 17]. The change rate of web pages is believed to be between a day to a year varying dramatically from site to site and object to object [7, 30]. The most popular objects have a higher rate of change than the others [7]. These changes can be modestly or significantly [16, 29, 30]. In [8], they study how often does a page change over all domains, and also for each domain (net-org, com, edu, gov) [8]. Wolf et al. show in [30] that 23% of web pages and 40% of commercial web pages change daily.

These statistics all show that a search engine index gets quickly out of date [30]. Keeping the index up-to-date regarding the web ever-changing huge content is a challenging task for search engines. A good search engine should be sensitive to changing data [8]. This tracking changes needs extra network resources [7]. To keep the extra load on network as minimized as possible, different strategies are applied that are studied in the following subsection.

3.2 Different Crawlers and Crawling Strategies

Different strategies are used for web crawling [24, 27]. In this paper, we divide crawlers into two main classes; general crawlers versus specific crawlers. We define general crawlers (also known as unfocused crawlers) as crawlers aiming at crawling all pages on the web while specific crawlers limit their targeted URLs based on focusing on a specific domain (Domain specific crawlers, Focused Crawlers), topic (Topic Specific Crawler), ontology (Ontology based crawlers), even a set of websites (Mobile crawlers), networks, or a geographical location. These crawlers can be distributed or run in parallel to distribute network loads.

In all of these crawlers, dealing with the huge amount of web documents and their frequent changes is an important challenge. Some crawlers apply simple re-visiting policies like uniform policy (re-visit all pages regardless of their change rates) or proportional policy (re-visit more the pages that change more) [7]. Although these are simple, they add extra network load (too often changing pages in proportional policy), and consider all pages on the web are worth the same.

In a more complex strategy, crawlers apply a selection policy to download next pages based on a number of different factors like importance of the web page, its change frequency, its intrinsic quality, its popularity in terms of links or visits and even its URL [7]. Some research work focus on modelling frequency of change of a web page by a Poisson process [8, 11, 9], or modelling web changes as a Renewal process [5, 6]. For example, Some formulate the crawling frequency problem based on stochastic marked points processes [30]. Web Caching and Hypertext Compression are other techniques applied to reduce network load caused by crawlers [20, 21].

3.3 Definition and Detection of Page Change in Crawlers

We define the possible changes in the content of the web by characterizing change events on the web as creations, updates and deletions [7]. A web page can be removed or created. These changes are only visible publicly on the web through updates to the pages that link to the removed or new pages. The page updates can be either minor or major. A minor update changes paragraph or sentence but keeps the page semantically almost the same. In the case of a major update, page changes completely and all references to its content are not valid any more.

To automatically judge if a page's content is changed, crawlers apply different techniques. Comparing pages on the basis of ASCII count of the new pages and comparing it with the old ones that had been downloaded, counting the number of URLs, number of keywords present in the web page, and checking the meta-data of the pages (like modification date) are some applied methods in crawlers to detect changed pages [27].

4. DETECTION OF IDENTICAL OR NEAR-IDENTICAL PAGES

Two documents with identical content are regarded as exact duplicates. However, in case of small dissimilarities between their content, they are identical to a remarkable extent. These documents are known as near duplicates [26]. For example, a few different words, different formatting but similar text, some typographical errors, plagiarized documents, different versions of a document, and different file types are considered as near-identical duplicates [26].

Broadly speaking, duplicate-detection systems have been developed for four types of document collections: web documents, files in a file system, e-mails, domain-specific corpora [25]. In web documents collections, different techniques are applied to find similar pages. Based on [25, 26], we propose the following classification of near-identical detection techniques in Figure **??**. As shown in Figure **??**, near-duplicate detection techniques have different feature-sets URL, Text Syntactic, Text Semantics, Structure, and Connectivity.

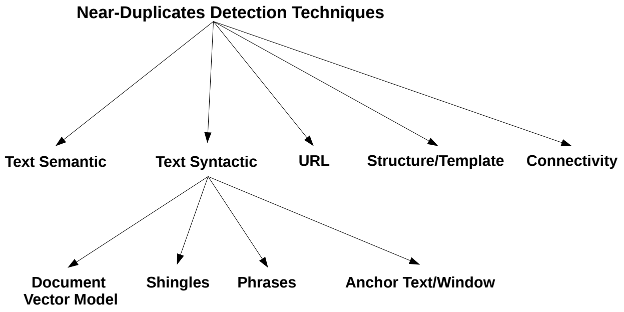


Figure 1: Classification of Near-Duplicate Detection Techniques

In URL-based techniques, instead of the page contents, only the URLs are used to find pages with similar contents. In connectivity-based approaches, information of linkage structure of the web is probed to find similar pages. The basic idea is that similar pages have similar incoming links. In structure-based techniques, the structural similarity of web pages are studied to identify schemas and templates of pages.

In text-syntactic-based approaches, analysing the syntax of a page content is the focus of approaches. In this analysis, we have different feature-sets representing a document. In Shingles-based technique, any sequence of k successive words is the feature set. For example, the k-shingles for ”a rose is a rose is a rose” with $k = 4$”, are ”a rose is a”, ”rose is a rose” and ”is a rose is”.

In Document-vector model, by using traditional IR techniques like stop-word removal, stemming, computing term-frequencies, etc., a document-vector is calculated to represent a document. In Anchor text, only the text surrounding an anchor is considered. In Phrases-based techniques, phrases in a page are detected and considered as terms in a document vector [25].

In Text semantic based approaches, instead of syntax of a page content, its semantics are considered. The Fuzziness-based and semantic-based graphs are two techniques applied in this category.

To apply these techniques on big data collections, we need to compress the feature set for fast comparisons. Mod-p shingles, Min-hash for Jacquard similarity of sets, Signatures/fingerprints over IR-based document vectors, and Checksums are among the techniques applicable as signature scheme for compressing the feature-set.

To measure similarity of two feature sets of two documents, different measurements are applied like Jacquard similarity or Levenshtein distance. In Jacquard, for two sets A and B, the similarity is defined as $\frac{|A \cap B|}{|A \cup B|}$. In Levenshtein distance, the number of inserts, deletes or substitutes of characters required to change one into the other are calculated. For example, the distance between ”sittin”, and ”sitting” is an insertion of ”g” at the end.

In our experiments, we define two documents as near duplicates by computing a Jacquard coefficient with a preset threshold of 0.9. We fix the K as the shingle size. Let S(A) be the set of shingles in A and let S(B) be the set of shingles in B. Compute $\frac{|S(D1) \cap S(D2)|}{|S(D1) \cup S(D2)|}$ (Jacquard coefficient). We extract text from web pages, calculate shingle set of each document and save the fingerprints of shingles. For each pair of documents, if the Jacquard coefficient exceeds threshold we considered them as changed documents.

5. RATE OF WEB CONTENT CHANGE

To study the change rate of the web content, different crawls of the web in different time stamps are required. In our studies, we considered three available options for a web crawl data set which includes different versions of the web. The following paragraphs describe these options.

CommonCrawl. The `CommonCrawl` corpus [19] contains petabytes of crawled data from the web including web pages, meta-data and text over several years. Access to the Common Crawl corpus that is hosted by Amazon Web Services is free. However, to compare at lease two versions of the Common Crawl collections in our system, we need a storage capacity of 500 TBs that was not available to us.

ClueWeb. The ClueWeb data set is a crawl of web aimed at supporting research on information retrieval and related human language technologies [1, 2]. It provides two versions of crawl of the web; ClueWeb12 [1] and ClueWeb09 [2]. The ClueWeb12 data set consists of 733,019,372 English web pages, collected in 2012 while ClueWeb09 was collected in 2010 with 1,040,809,705 web pages, in 10 languages. The time difference in versions is about two years and they apply different processes and mechanisms in crawling which creates difficulties for change comparison.

Fedweb. The FedWeb data set [3, 15] is designed for research in Federated Web Search. The authors provide two version; FedWeb'13 and FedWeb'14 [3]. Each version consists of the top-10 search results of posing sampled queries, as well as a set of test topics, on about 150 search engines [3]. Regarding the size of the collections and the only one year time difference of the two versions, we chose FedWeb for our analysis.

5.1 Change rate of websites in FedWeb

In [14, 13], the authors crawl around 150 websites in two versions of crawls from 2013 and 2014. To analyse the change rate of websites in FedWeb collections, we considered the new URLs among the results for a same query during 2013 and 2014 crawls. We also investigated the change in content for similar URLs. The results of this comparison are shown in Figure 2. In this figure, the number of queries that could return results after their submissions to each website and therefore considered in our evaluation is also depicted.

5.2 Change rate of categories in FedWeb

The FedWeb 2013 Data Collection consists of search results from around 150 web search engines in 24 categories covering a wide range of domains [14, 13]. These categories and the number of websites from each category are mentioned in details in Table 1. The change rates for websites, shown in 2, are grouped for each mentioned category in Table 1. The change rates of categories with their corresponding error bars are depicted in Figure 3.

Table 1: Categories Count

Category	Count	Category	Count
Academic	17	Local	1
Audio	6	News	12
Blogs	4	Photo/Pictures	13
Books	4	Q&A	7
Encyclopedia	5	Recipes	5
Encyclopedia	4	Shopping	9
Games	6	Social	3
General	10	Software	3
Health	13	Sports	7
Jobs	5	Tech	8
Jokes	2	Travel	2
Kids	9	Video	14

5.3 Change Rate Analysis

As a result of our experiments to analyse the web change, it is interesting to pint out that the change rate differs from website to website and category to category. Although some categories are not represented thoroughly in FedWeb collection (refer to Table 1), we can still draw some conclusions. As shown in Figure 3, some categories like *Jobs, Shopping, Sports and Blogs* show near 100 percent change while in *Travel wikis and Kids* websites, we see much less changed content. Even in each category, websites have differences in change rates. These differences are depicted as error bars in Figure 3.

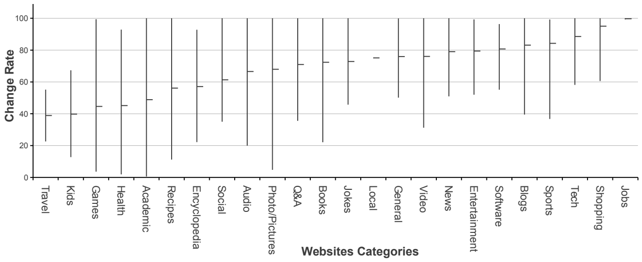


Figure 3: Change Rate of Categories in FedWeb

In general, we can claim that the web content in FedWeb collection changed in average 40 percent. However, the change rate is subjective to domains and websites.

6. SOLUTION FOR WEB CONTENT MONITORING

As the first step in monitoring change in the web content, we need to have a first complete crawl at hand. As investigated in [?], the most efficient approach for focused web harvesting is a Combined-List-Feedback approach. In [?], the Combined-List-Feedback approach is defined as using a predefined list of words and also the extracted content from query search results. We apply the same technique for having the first data collection.

After a predetermined time that is set to two weeks in our experiments, we try for the second harvest. We re-harvest documents with the same queries to have a baseline for detecting changes. This second try of harvesting is referred to as *SecondCrawl*. To detect changes, we define a change as retrieving a new document or change in content of a previously retrieved page. Then, we detect and count changes in the *SecondCrawl*. We compare the contents of two pages based on the mentioned technique in Section 4.

In this work, our goal is to implement approaches that harvest only the changed pages. Our proposed approaches are based on expanding the seed query (given entity). To expand the seed query, we favour the terms that, in general, generate fewer duplicates, big samples, result in more changed and fewer unchanged pages. To find these terms, we employ different strategies. Analysing the extracted results, external corpora, and list of words for the seed query expansion are among these strategies. The suggested approaches based on these strategies are listed below.

6.1 Most Frequent From New Documents

In this method, the terms used to reformulate queries are selected from the previously retrieved content (feedback-based) that is also detected as changed. Among the terms

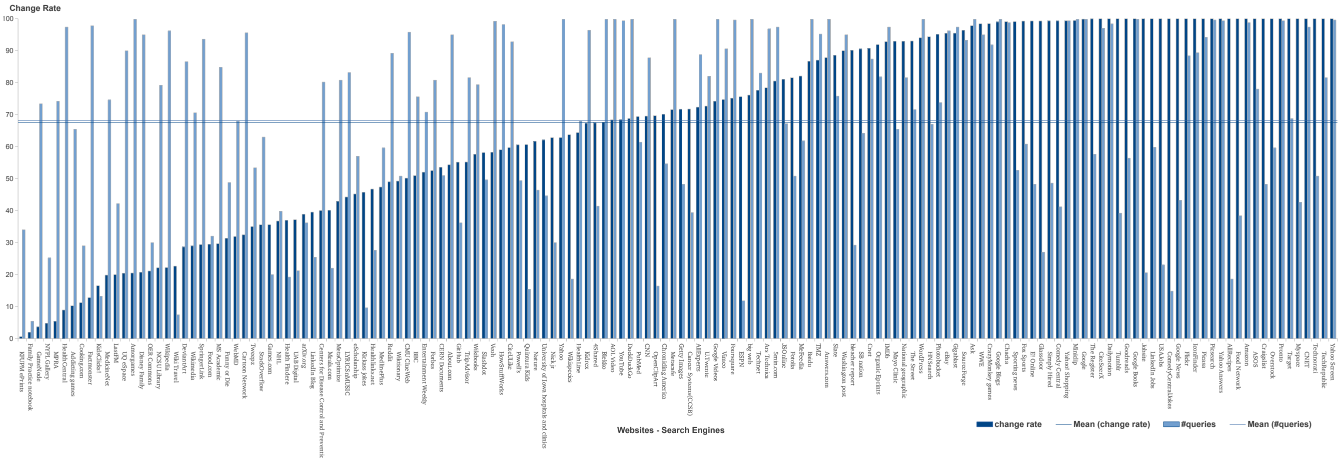


Figure 2: Change Rate of Websites in FedWeb

extracted from these changed documents, the ones with higher frequencies have a higher chance in returning bigger samples. This method submits the seed query to the test search engine, detect new and changed documents. 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. For example, for the given entity “PhD Comics”, the term “graduate” appears as the most frequent term in the returned changed documents. The next step is submitting “PhD Comics”+“graduate”. This approach is referred as the *MostFreq* approach.

6.2 Least Frequent From New Documents

The *LeastFreq* approach performs on the same basis as *MostFreq* approach but instead of selecting most frequent terms to reformulate queries, the least frequent ones are selected.

6.3 Combined List and Feedback From New Documents

The most and least frequent terms represent extreme cases increasing chances of consecutively bigger samples and fewer duplicates. To have a balanced approach, in [**?**] the suggested calculates a specific frequency and creates a list of terms with this frequency from an external corpus (a list-based approach). Then, the terms from this list that also appear in the previously retrieved content (feedback-based approach) are selected to reformulate the seed query [**?**]. This predetermined frequency can be calculated through Formula 1 if search engine size and number of matching documents to the seed query are known. We can estimate the search engine size if it is unknown [**?**]. The document frequencies of terms are pre-computed from an external corpus (ClueWeb data set).

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

$$|Sample^{Size}| = \frac{l * size^{SE}}{|R^{Coll}|} \mid (|R^{Coll} \cap Sample^{Size}| = l)[?] \qquad (1)$$

For example, with $size^{SE} = 10^9$, the number of English documents in ClueWeb as 5×10^8, $l = 100$ and $|R^{Coll}|$, the number of matching documents to a given query, for the seed query as 4×10^5, through $\frac{100}{10^9} = \frac{4\times 10^5}{10^9} * \frac{x}{5\times 10^8}$ we can calculate the $\Longrightarrow x = 125000$ in which the x represents the frequency used for selecting terms from the external corpus [**?**].

To apply this approach to retrieve changed documents, we consider the changed documents as the previously retrieved content. Therefore, if the terms from the list appear among the retrieved changed content, they will be used for query reformulation. This approach is referred to as *Combined* approach.

6.4 FedWeb1

In this method, we analyse different versions of crawls of the web created in different time points. The goal of this analysis is to detect changed documents and find the list of most representative words of this changed documents set.

In this work, we analyse two different versions of FedWeb to find a list of words that are more common in changed documents than unchanged ones. To expand the seed query in *FedWeb1* approach, we choose a term to be added to the seed query from a list of words that are representative of the changed documents in the FedWeb collection.

To find the list of most representative terms for changed documents, we apply Naive-Bayes classifier. We classify the documents into changed and unchanged documents by comparing their contents through Shingle-Jacquard text comparison technique (described in Section 3.3). We use these classified documents to train our Naive-Bayes classifier. From this classifier, we get the list of representing terms for each category [18, 28].

To be accurate, we apply a Complement Naive Bayes classifier that seeks to maximize term weights on the likelihood that they do not belong to any other class [18, 28]. In this classifier, the documents are represented as vectors. Each document vector has a label. By defining a smoothing parameter for all words in the vocabulary, and applying the TF-IDF transformation and L2 length normalization, we can assign each term with a corresponding weight [18, 28].

After calculating these weights for all the terms in documents that belong to the changed documents category, the terms are ordered based on their weights. The top terms of this list are used to form queries to retrieve changed documents in our experiments. This method is referred as *FedWeb1* approach.

7. EXPERIMENTS AND RESULTS

In our experiments, we run the proposed approaches on a *test search engine.* We detect changed documents based on our *change definition* and analyse the results.

7.1 Experiments Settings

Test Search Engine We test our approaches on a real search engine. In these experiments, Google as the biggest web search engine is considered as our test search engine. We believe Google is one of the most representative collections of the web including a wide range of domains and a large number of entities. Although we targeting Google, there is no limitation on applying these approaches on other websites as far as they provide keyword-search.

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.

Change Definition In our experiments, we define change on the web as finding new URLs or similar URLs with changed content. If there is change in the set of returned results for a same query, submitted in different times, we do not consider it as a change. We consider this kind of change more specific to search engines and their ranking algorithms. Therefore, this change is not presented in our results.

Evaluation Metric To assess different approaches for different entities with different sizes of matching document and change rates, we consider the percentage of the number of new documents for each entity as the comparison metric.

Fixed l In our test search engine, Google, the number of returned results is not fixed and vary from 200 to 500 even for the same query but at different times. 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.

7.2 Results

In this section, the results of applying the approaches introduced in Section 6 to the test entities (Subsection 7.1) are presented. To establish a comparison baseline, we send the exact same queries from the previous crawl and refer to it as *SecondCrawl* in the graphs. The idea is to see the amount of changes occurred on the web for the exact same information needs.

With this baseline, we apply the proposed approaches in Section 6 on the test search engine and compare the amount of retrieved changed documents. Labelling a page changed or not is based on the given *change definition* in Subsection 7.1.

In Figure 4, the performances of all the introduced approaches in Section 6 to retrieve new documents are compared. As shown in this figure, the FedWeb approach outperforms the other approaches by **20 percent**. In the next section, we study the reasons behind these performances.

8. DISCUSSIONS AND ANALYSIS

As shown in Figure 4, the FedWeb approach outperforms the other proposed approaches. In the following, we discuss the reasons behind this performance and also our other findings.

As the first reason, in the FedWeb approach, we send the representative set of words from changed content in the FedWeb. It seems these words are indeed good representatives of changed content.

From [**?**], we know that for improving coverage in a focused harvesting task, the approach needs to have more returned results (bigger samples) and fewer duplicates among these samples. In another word, for a better coverage, we need queries which lead to more unique documents. In retrieving changed documents, the methods should consider the same principle but for changed content. In fact, the goal is more coverage on changed content. Therefore, we need to have more new documents in each query submission, while reducing the number of duplicates among these new documents. As shown in Figure **??**, the FedWeb approach is the one satisfying the requirements in retrieving more unique documents considering 5a, and 5b figures. We also show in figures 5c and 5d that FedWeb approach outperforms other approaches in returning not only more documents but more unique and new documents too.

It is important to note that the approaches also return documents that were not retrieved in our previous crawls. This will lead to better performance of LeastFreq approach. This is observed in Figure 6. In this approach, we submit queries formed by adding the least frequent terms among the already retrieved documents to the seed query. As these terms are less frequent, they return documents less likely to be covered by other approaches.

We also noticed that change and its rate are subjective to each topic and domain. Although, we noticed differences

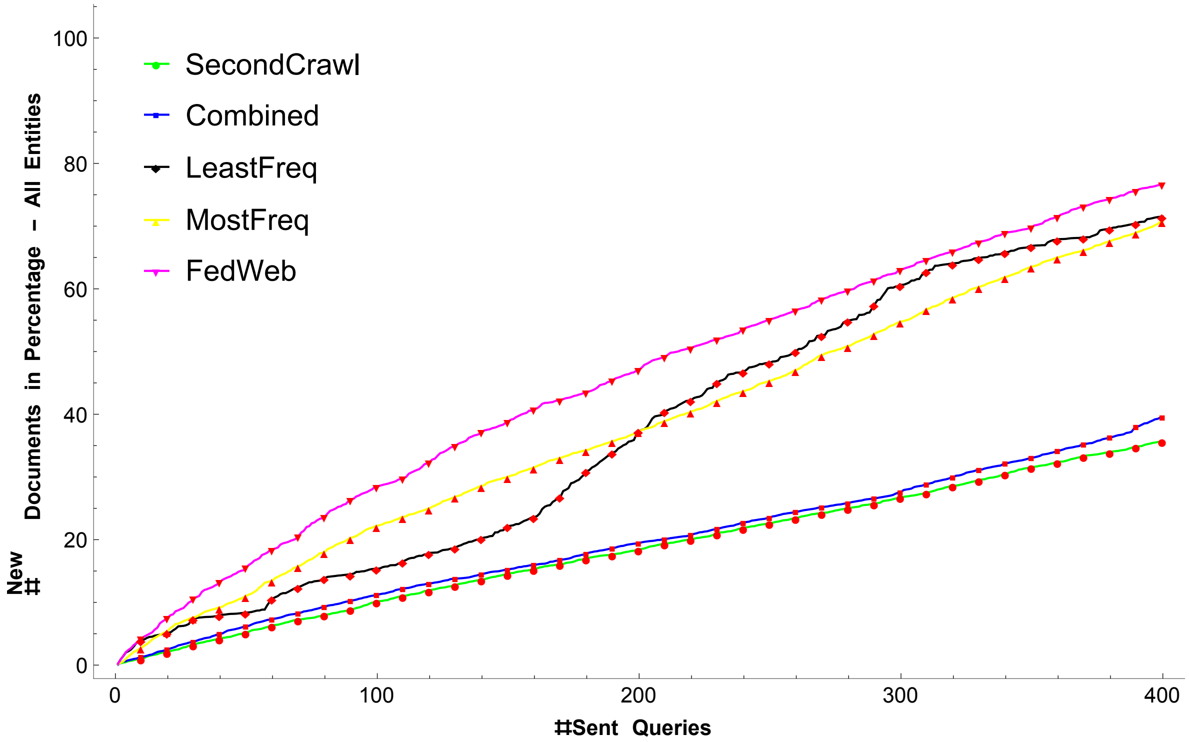
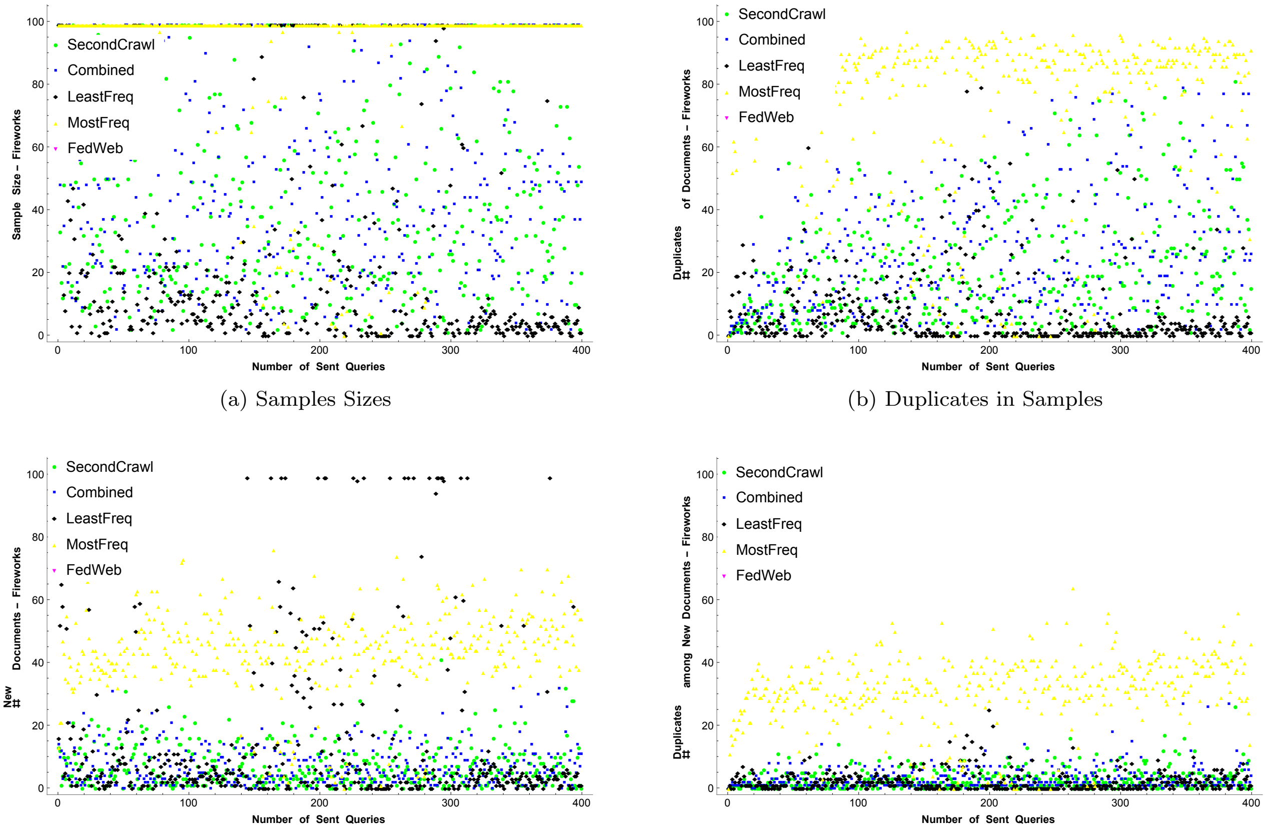


Figure 4: All New Documents for All Entities in Test Set



(a) Samples Sizes

(b) Duplicates in Samples

(c) New Documents in Samples

(d) Duplicates among New Documents

Figure 5: Performance of Different Approaches for an Example Entity

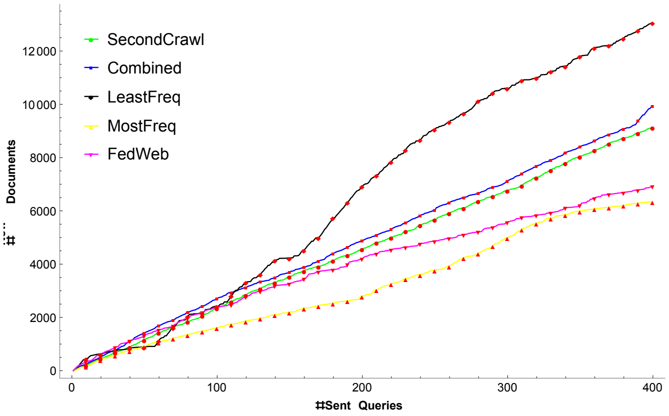


Figure 6: All New Documents for an Entity with Incomplete Coverage

among the number of retrieved new documents among different entities in the test set, the FedWeb approach was always among the top performing approaches.

As another finding of this work, we noticed a big change in the top-100 returned results by Google. In the *Second-Crawl*, we submitted the exact queries to Google and faced in average around 50 percent difference among the returned results.

9. CONCLUSION AND FUTURE WORK

Conclusion. As mentioned in the Introduction Section, our goal is to find efficient methods that can return the changed web data for a given topic. To do so, first we analysed the FedWeb data set to study the change rates and their differences among different websites and categories. We showed that FedWeb data changed in average 40 percent cross different websites. We also noticed that this change is highly subjective to domains and categories and varies from domain to domain. Our findings correlates with the results presented in [7, 30] that web data hange rate is subjective to domains, topics, websites and categories. It also shows again that the web is dynamic and changing rapidly. This change rate information is important to choosing the most appropriate time to re-harvest data sources to get new information.

To detect these changes, we introduced four different approaches to efficiently harvest all the new and changed documents matching a given entity by querying a web search engine. Among these 4 approaches, the FedWeb approach outperformed the others. The FedWeb approach, that is based on a set of terms representative of changed documents in FedWeb data set, produced more new documents and fewer duplicates. This approach performed the best with at least 20 percent difference from the others in the rate of changed documents detection.

Future Work. In this work, because of limitations on the resources, we could not include ClueWeb or Common Crawl data collections. If these limitations are uplifted, we can consider the two following approaches as future work. The first one is based on analysing two different versions of a thorough web crawl and assigning weights to websites representing their corresponding number of changed pages. With these weights at hand, we can select the terms to expand seed query from documents that belong to websites with higher weights. In this method, all websites should have a weight which needs to analyse big indexes and collections.

The second method of future work is based on analysing two different versions of a thorough web crawl and training a classifier for terms and changed documents. This trained classifier can be used for predicting the number of changed pages a query can result. Therefore, we can select terms with higher chances of resulting in more changed pages to expand the seed query.

In this work, we mention the application of calculated change rate for websites and domains for selecting the best time to run change detection methods for a given topic. The actual application of these change rates can be also studied as a future work.

10. REFERENCES

[1] The clueweb09 dataset, 2015.

[2] The clueweb12 dataset, 2015.

[3] Fedweb greatest hits, 2015.

[4] Ricardo Baeza-Yates, Carlos Castillo, and Felipe Saint-Jean. Web dynamics, structure and page quality. In M. Levene and A. Poulovassilis, editors, *Web Dynamics*, pages 93–109. Springer Verlag, 2004.

[5] Brian E. Brewington and George Cybenko. How dynamic is the web? In *Proceedings of the 9th International World Wide Web Conference on Computer Networks : The International Journal of Computer and Telecommunications Netowrking*, pages 257–276, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.

[6] Brian E. Brewington and George Cybenko. Keeping up with the changing web. *IEEE Computer*, 33:52–58, 2000.

[7] Carlos Castillo, Dr. Alistair Moffat, and Dr. Gonzalo Navarro. Effective web crawling. Technical report, 2004.

[8] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 200–209, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[9] Junghoo Cho and Hector Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 117–128, New York, NY, USA, 2000. ACM.

[10] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Trans. Internet Technol.*, 3(3):256–290, August 2003.

[11] E.G. Coffman, Zhen Liu, and Richard R. Weber. Optimal robot scheduling for web search engines, 1997.

[12] Maurice de Kunder, 2015.

[13] Thomas Demeester, Dolf Trieschnigg, Dong Nguyen,

and Djoerd Hiemstra. Overview of the TREC 2013 federated web search track. In *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*, 2013.

[14] Thomas Demeester, Dolf Trieschnigg, Dong Nguyen, Djoerd Hiemstra, and Ke Zhou. Overview of the TREC 2014 federated web search track. In *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*, 2014.

[15] Thomas Demeester, Dolf Trieschnigg, Ke Zhou, Dong Nguyen, and Djoerd Hiemstra. Fedweb greatest hits presenting the new test collection for federated web search. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, New York, NY, USA, 2015. ACM.

[16] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. Rate of change and other metrics: a live study of the world wide web, 1997.

[17] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. Rate of change and other metrics: A live study of the world wide web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, USITS'97, pages 14–14, Berkeley, CA, USA, 1997. USENIX Association.

[18] The Apache Software Foundation. The apache mahoutâĎć, 2015.

[19] The Common Crawl Foundation. The common crawl dataset, 2015.

[20] Zvi Galil and Nimrod Megiddo. A fast selection algorithm and the problem of optimum distribution of effort. *J. ACM*, 26(1):58–64, January 1979.

[21] Toshihide Ibaraki and Naoki Katoh. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, Cambridge, MA, USA, 1988.

[22] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[23] Lipyeow Lim, Min Wang, Sriram Padmanabhan, Jeffrey S. Vitter, and Ramesh C. Agarwal. Characterizing web document change. In *Proceedings of the Second International Conference on Advances in Web-Age Information Management*, WAIM '01, pages 133–144, London, UK, 2001. Springer-Verlag.

[24] Swati Mali, Vjti Mumbai, B. B. Meshram, and Vjti Mumbai. Focused web crawler with page change detection policy.

[25] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 141–150, New York, NY, USA, 2007. ACM.

[26] SY Mudhasir, J Deepika, S Sendhilkumar, and GS Mahalakshmi. Near-duplicates detection and elimination based on web provenance for effective web search.

[27] Rajender Nath, Naresh Kumar, and Sneha Tuteja. A survey on reduction of load on the network. In Rajkumar Buyya and Sabu M. Thampi, editors, *Intelligent Distributed Computing*, volume 321 of *Advances in Intelligent Systems and Computing*, pages 239–249. Springer International Publishing, 2015.

[28] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.

[29] Crai E. Wills and Mikhail Mikhailov. Towards a better understanding of web resources and server responses for improved caching. In *In Eighth International World Wide Web Conference*, pages 153–165, 1999.

[30] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *Proceedings of WWW '02*, pages 136–147, 2002.