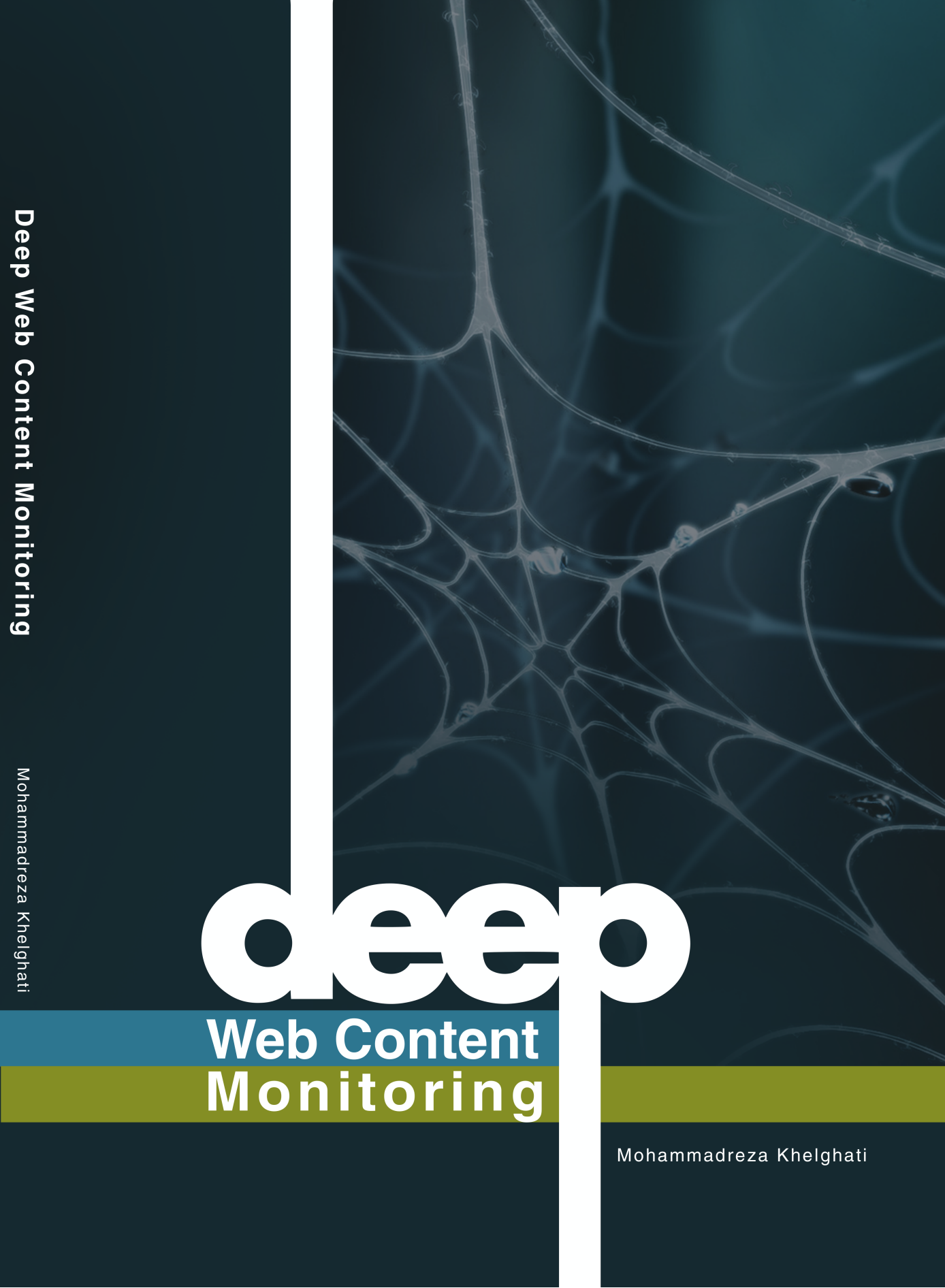

Deep Web Content Monitoring
Mohammadreza Khelghati
deep
Web Content
Monitoring
Mohammadreza Khelghati

Deep Web Content Monitoring

S. Mohammadreza Khelghati

Graduation committee:

Chair:	Prof. dr. P.M.G. Apers
Supervisor:	Prof. dr. P.M.G. Apers
Co-supervisor:	Dr. ir. Djoerd Hiemstra
Co-supervisor:	Dr. ir. Maurice van Keulen

Members:

Prof.dr. Pierre Senellart	Télécom ParisTech, France
Prof.dr. T.W.C. Huibers	University of Twente, The Netherlands
Prof. dr. Franciska de Jong	University of Twente, The Netherlands
Dr. Andrea Cali	Birkbeck, University of London, UK
Dr. Gianluca Demartini	Univeristy of Sheffield, UK

CTIT Ph.D.-thesis Series No. 16-391
Centre for Telematics and Information Technology,
University of Twente
P.O. Box 217, NL – 7500 AE Enschede

SIKS Dissertation Series No. 2016-31
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN 978-90-365-4123-7
ISSN 1381-3617 (CTIT Ph.D. thesis Series No. 16-391)
DOI: 10.3990/1.9789036541237
http://dx.doi.org/10.3990/1.9789036541237

Cover design: Sanaz Khelghati and Elham Toutouni

Deep Web Content Monitoring

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
Prof.dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday June 2nd, 2016 at 12:45

by

Mohammadreza Khelghati

born on the 27th of June, 1985
in Zanjan, Iran

This dissertation has been approved by:

Prof. dr. P.M.G. Apers (supervisor)
Dr. ir. Djoerd Hiemstra (co-supervisor)
Dr. ir. Maurice van Keulen (co-supervisor)

تقدیم به پدر

مادر

و خواهر عزیزتر از جانم

CONTENTS

CHAPTER 1

INTRODUCTION

In this chapter, our motivations behind writing this book, the raised research questions and the general structure of the book are described.

Data is one of the keys to success. Whether you are a fraud detection officer in a tax office, a data journalist [48, 65] or a business analyst, your primary concern is to access all the data that is relevant to your topic of interest. In any of these roles, an in-depth analysis is infeasible without a comprehensive data collection. Therefore, broadening the coverage of available information is a vital task. In such an information-thirsty environment, accessing every source of information is valuable. This emphasizes the role of the web as one of the biggest and main sources of data [94]. The web has an abundance of valuable public data, continuously produced and shared [52]. The web data has been used for a wide range of applications (i.e. business competitive intelligence [143] or crawling social web) to understand complex economical and social phenomena [52].

Nowadays, the most common approach to look for information on the web is posing queries on general search engines such as Google [61, 93, 94]. However, none of these search engines cover all the indexed web data [3, 19]. They also miss data behind web forms. This data that is hidden from search engines behind search forms is defined as *hidden web* or *deep web*. It is estimated that

deep web contains data in a scale several times bigger than the data accessible through search engines called *surface web* [15,70,115].

In accessing deep web data, finding all the interesting data sources and harvesting them through their own interfaces could be a difficult, time-consuming and tiring task. Considering the huge amount of information that might be related to one's information needs, it might even be impossible for a person to cover all the deep web sources of his interest. Therefore, there is a great demand for applications that can facilitate the access to this big amount of data that is locked behind web search forms.

Of course, the availability of an up-to-date crawl including the deep web will definitely facilitate collecting all relevant information on a given entity. However, given the software and hardware requirements of keeping an up-to-date crawl of the whole web, this seems to be still impracticable even for big organizations like Google [3,104,105,133].

Consequently, to access web data, a journalist has to resort to using general search engines, direct querying of deep web or both of these two approaches. Considering all these cases, the laborious work of querying, navigating through results, downloading data, storing it and tracking its changes can significantly benefit from automatic web data access approaches [51,65,142].

With this thesis, we aim to make a real-world automatic web data access approach that provides users with collections of *all* the relevant data to their desired topics.

1.1 MOTIVATION

In accessing web data through either general search engines or direct querying of deep web sources, the laborious work of querying, navigating results, downloading, storing and tracking data changes is a burden on shoulders of users. To decrease this intensive labor work of accessing data, (semi-)automatic harvesters have a crucial role. A web harvester is a software program enabling web harvesting. Web harvesting (also known as web data/information extraction and web mining) is defined as automatically retrieving web pages of interest, extracting information from them and storing and integrating data [39,57]. Through these steps, the web harvesting process extracts structured data from unstructured and semi-structured data.

As a web user, while utilizing our cognitive abilities, we look for information of our interest by navigating through websites, web forms and web pages. We can easily find search boxes and search buttons, navigate through results by

clicking on the next button, distinguish between advertisements and search results, and locate the information of our interest in browsed pages. Now imagine you need to find that information in a Persian website and you have no knowledge of the Persian language. This is analogous to computers making it difficult for them to understand the web. If there was only one way and one language for programming websites, computers would have faced fewer difficulties in understanding them. The diversity of website programming techniques, coding styles, website domains, languages and other features of websites makes it harder to have one approach that can be applied to all these different settings. This is hard as a harvester should be configured to understand all these diversities. Currently, harvesters typically work by one configuration per website and need minor or major changes to be still applicable to the same website when it changes or be applied to other websites and domains.

Targeting different websites and domains increases the need to have a general harvester which can be applied to different settings and situations. Despite extensive research on advancing harvester technology [15, 25, 39, 47, 51, 58, 64, 70,74,92,105,116,126,129,136,139], there are still many open challenges to have a fully automated harvester that finds websites of interest, queries them, navigates through search results, extracts information, aggregates all data, filters out noise and presents users with what they asked for. For a harvester, in each of these mentioned tasks, there are unresolved challenges (a complete list of these issues are discussed in Chapter 2). This thesis targets some of these challenges with the goal of having a general web harvester that can automatically collect all the information relevant to a topic of interest.

1.2 Research questions

The main goal of this thesis is to take several steps towards a web data access approach that automatically queries websites, navigates through results, downloads data, stores it and tracks its changes. To reach this goal, the thesis centers on the following research questions (RQs).

RQ 1: How can we improve data coverage of harvesters given the limitations imposed by search engines, limited resources of harvesters and adherence to politeness policy?

Although using a fully automatic general harvester facilitates accessing web data, it is not a complete solution to achieve a thorough data coverage of a given topic. Some search engines, in both surface web and deep web,

restrict the number of requests from a user or limit the number of returned results presented to him. These limitations make it harder for harvesting all the related documents to a given topic. Therefore, it is vital to find methods that, given these restrictions, can still achieve the best coverage in harvesting data for a given topic. This RQ is answered in Chapter 4. These methods can also improve the efficiency of access approaches.

RQ 2: To improve the efficiency of a harvester, how can we estimate the size of a harvested website?

To reduce the costs of harvesting a website regarding the number of submitted requests, it is important to know the status of a harvesting process regarding the amount of downloaded data as the process goes on. Harvesting processes continue till they face the posed query submission limitations by search engines or consume all the allocated resources. To prevent this undesirable situation, a mechanism should be applied to stop crawling process wisely. This means to make a trade-off between the resources being consumed, limitations and the percentage of a deep website that is crawled. To do so, one of the most important factors is to know the size of the targeted source. This prevents a harvester from sending unnecessary requests and a user can choose the best time to stop the process avoiding unnecessary consumption of resources. This is especially important in harvesting websites that hide the true size of their residing data. For these websites, the harvester is incapable of deciding if a website is fully harvested unless through a very costly harvesting process. Therefore, it is important to know the amount of data coverage by a harvester as a harvesting process goes on to decide when the process should be stopped. This RQ is answered in Chapter 3.

RQ 3: Given the ever-changing nature of the web, how can we keep the collection of the related documents to given topics of interest up-to-date and monitor it over time?

Having an efficient deep web data crawler at hand, it is of our interest to be aware of the changes occurring over time in existing data in deep web data repositories. This information is helpful in providing the most up-to-date answers to information needs of users. The fast evolving web adds extra challenges for having an up-to-date data collection. If we assume that all the relevant information to a given topic is harvested, what is the best time to reharvest data sources to get new information? How can we get new relevant

data to a given topic without re-harvesting all the previously downloaded documents? Considering the costly process of harvesting, it is important to find methods which facilitate efficient re-harvesting processes. This RQ is answered in Chapter 5.

RQ 4: What are the features to consider while designing and developing a general access approach that can be applied to a wide range of websites, domains and tasks? How can we prioritize implementations of these features?

As the first step in designing an automated web data access approach (web harvester), we should decide about which features to include in our system. To do so, we need to know the state-of-the-art of automatic access approaches. We should look for available design frameworks or guidelines. It is important to know how to configure a harvester so that it can be applied to different websites, domains and settings. To automatically harvest our data of interest from any website, we need a harvester that does not require any site specific configurations. How to design and develop such a general web harvester is the key question of this research topic. This question is addressed in Chapter 6.

A key element to improve a system is the availability of a comparison metric. For a general web harvester, we lack a well-defined comparison metric. It is important to define a metric which enables thorough comparison of features in harvesters. Due to the existing diversity of websites, domains, harvesting techniques and tasks, comparing harvesters only by the amount of collected data does not reveal enough information on their performances. To have a completely automated web harvester, we need to define precisely where the current harvesters stand and on what dimensions they need to be improved. A thorough comparison metric can help us to reach this goal. This issue is also addressed in Chapter 6.

1.3 Contributions

The findings from this study make several contributions to the current literature. First, we study the web as a source of websites from both deep web and surface web and categorize data access approaches for these websites. Secondly, we present challenges that these web access approaches face in one big picture. Thirdly, a thorough literature study is done on web harvesters as one of the methods to access web data. Different web harvester techniques are described and categorized.

These studies enhance our knowledge of web harvesters and suggest a new framework to design a general web harvester. One of the findings is a new factor called *harvestability factor (HF)* which can be used as a comparison metric for web harvesters and websites regarding their capabilities of harvesting and being harvested consecutively. This is a new factor that empowers designers of websites and web harvester with better metrics to measure where their products stand regarding harvesting capabilities.

A key strength of the present study was its focus on enhancing efficiency in web harvesters. This is followed by introducing more accurate techniques for estimating the size of a website collection. The current findings add to a body of literature on the size estimation of non-cooperative data collections. This finding of the thesis can be used to design more intelligent web harvesters which can decide whether to continue or stop a harvesting process as it proceeds based on the amount of downloaded data and the estimated size of targeted collection. This intelligent decision-making process results in designing more efficient harvesting processes.

The present study makes noteworthy contributions to the topic of focused web harvesting. Our findings take steps towards a complete coverage of data in focused web harvesting and intend to empower normal users to access all the relevant documents to their topics of interest. We also introduce approaches to keep the downloaded collection of documents up-to-date. Instead of re-harvesting all relevant documents, we provide efficient methods to download only the changed and new documents.

To serve as a base for future studies, the developed software used for running experiments of this thesis is made publicly available [79].

Example applications This research has several practical applications. During this research, the findings were applied in a wide range of applications. From business research to competitions among companies, we tested and applied suggested approaches and research findings.

- Cooperation with *Technology Management and Supply* group at the University of Twente[1] to harvest information about game development in research on optimizing technological and organizational recombination,
- Cooperation with *MyDataFactory*[2] (a Dutch software company) on harvesting product data for data cleaning purposes,

[1] `https://www.utwente.nl/bms/tms/`
[2] `http://www.mydatafactory.com/`

- Cooperation in *Freedom of Information Document Overview*[3] *(FIDO)* golden demo to extract entities from web sources, and

- Cooperation with *WCC*[4] to extract job vacancies and applying a framework to design a general web harvester.

To have a better understanding of some of the other potential applications of this research, please refer to Section 2.2.

1.4 THESIS STRUCTURE

This thesis starts with the introduction of deep web, surface web and access approaches to the available data in websites belonging to either of them in Chapter 2. Chapter 2 also discusses challenges for all steps of accessing data.

Chapter 3 proposes an accurate technique for estimating the size of a non-cooperative data collection. Knowing about the size of a targeted collection helps to understand the status of a harvesting process as it proceeds and accordingly, leads to prevent unnecessary consumption of resources that results in more efficient web harvesters.

In Chapter 4, we focus on moving towards a complete data coverage in focused web harvesting. In this chapter, the challenges for retrieving all the relevant data to a given topic are discussed and different techniques are suggested to address these challenges.

Chapter 4 is followed by proposing a number of techniques to make efficient re-harvesting of relevant data to a given topic in Chapter 5. This is required for detecting changed data on the web for a given topic of interest.

Chapter 6 describes a general guideline for designing general-purpose harvesters. In this chapter, a new metric is introduced to measure and compare the performances of different web harvesters.

Chapter 7 is dedicated to draw main conclusions and future steps of this research. The relation of these chapters are shown in Figure 1.1.

[3] http://www.taalmonsters.nl/projects/Fido
[4] https://www.wcc-group.com/

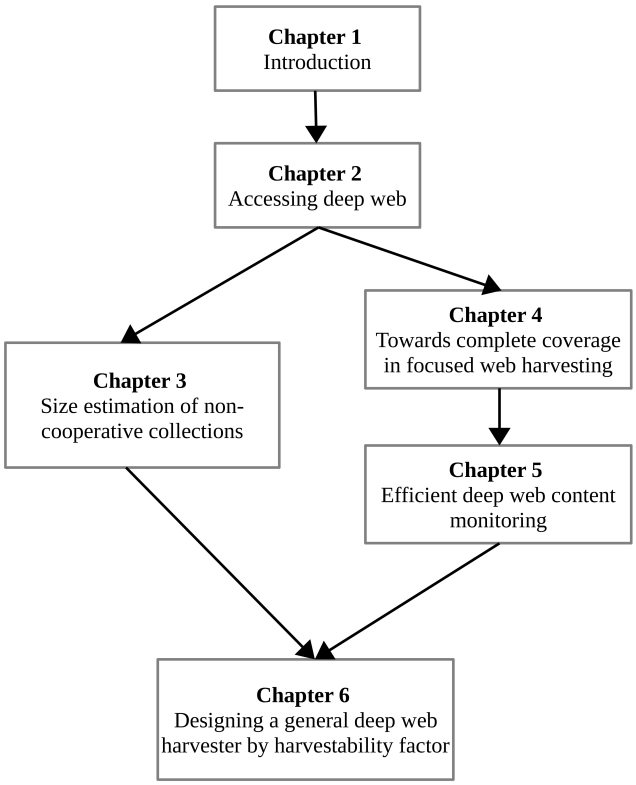


Figure 1.1: Thesis chapters

CHAPTER 2

ACCESSING THE DEEP WEB

What is deep web and how is it different from surface web? How can we access deep web data? What is the state-of-the-art in deep web access approaches? What are web data access challenges and which of them are targeted in this work?

The content of this chapter is based on [78].

2.1 WHAT IS DEEP WEB?

The most commonly applied method by general search engines like Google and Bing to discover web content is to follow every link on a visited page [43, 105]. The content of these pages is extracted, analyzed and indexed for being matched later against user queries. This indexed content is defined as the *surface web* [17, 43, 105]. By following the links in visited pages, we miss a part of the web that is hidden behind web forms. To access this part, a user should submit queries through web forms (Figure 2.1). As this part of the web is invisible or hidden from general search engines, it is called the *hidden* or *invisible web* [17, 43]. However, by applying a number of techniques, the invisible web can be accessible to users. Therefore, it is also called the *deep web*. Deep web refers to the hidden content behind web forms that standard crawl techniques cannot easily access [38, 43, 71].

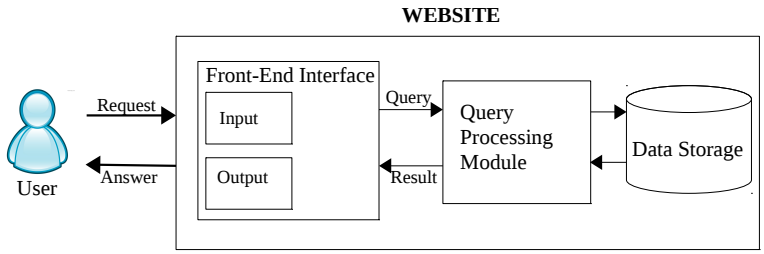


Figure 2.1: Accessing a website in deep web by a user [144]

Deep web content includes dynamic, unlinked, limited access, scripted or non-HTML/text pages residing in domain specific databases and private or contextual web [43]. This content is in the form of structured, unstructured or semi-structured data. Deep web content is diversely distributed across all subject areas from financial information and shopping catalogs to flight schedules and medical research [4]. In "Crawling deep web entity pages" [71], two different types of deep websites are introduced: document-oriented and entity-oriented. Yeye *et al.* [71], defined document-oriented deep websites as websites containing mostly unstructured text documents such as Wikipedia, Pubmed [121] and Twitter. On the other hand, entity-oriented deep websites are considered to contain structured entities: almost all shopping websites, movie sites and job listings [71]. Entity-oriented deep websites are suggested to be very common and represent a significant portion of deep websites.

The following section discusses the reasons behind the importance of deep web harvesting. The state-of-the-art in accessing deep web data is discussed in Section 2.3. In Section 2.4 lists and explains the challenges to have a harvester that fully automatically finds websites of interest, queries them, navigates results, extracts information, aggregates all data and presents it to users. In Section 2.5, only the challenges targeted in this research are discussed.

2.2 WHY DEEP WEB?

Accessing more data sources is not the only reason that makes deep web data interesting for users, companies and researchers. In the following paragraphs, a number of additional reasons that make deep web more attractive are mentioned.

Huge amount of high quality data In a survey performed in 2001, it was estimated that there are 43,000 to 96,000 deep websites with an informal estimate of 7,500 terabytes of data compared to 19 terabytes of data in the surface web [17, 38, 115]. In another study, it was estimated that there were 236,000 to 377,000 deep websites having an increase rate of 3-7 times in volume during 2000-2004 period [70, 115]. Adrian *et al.* [9] calculated that deep web contains more than 450,000 web databases that mostly contain structured data (relational records with attribute-value pairs). They showed that these structured data sources have a dominating ratio of 3.4 to 1 versus unstructured sources [70]. A significant portion of this huge amount of data is estimated to be stored as structured/relational data in web databases [9, 17, 71]. More than half of deep web content resides in topic specific databases [17, 33]. This makes search engines capable of providing highly relevant answers especially to subject-specific information needs.

Bergman [17] also mentioned that more than ninety-five percent of deep web data sources are publicly accessible and require no fees or subscriptions. The rest of these sources are partially or completely subscription/fee-limited which is not a large part of deep web.

Search engines fail in satisfying some of our information needs Trying queries like "what is the best fare from New York to London next Thursday" [4] or "count the percentage of used cars which use gasoline fuel" [144] on search engines like Google, Bing, MSN and others shows that these search engines are missing the ability to provide users with answers to their information needs. For such a query, general search engines do not provide users with the best websites through which users can find the data they are looking for.

Access mission-critical information Through a deep web harvesting process, mission-critical information (e.g. information about competitors, their relationships, market share, R&D, pitfalls, etc.) existing in publicly available sources becomes available for companies to create business intelligence for their processes [38]. Such access can also enable viewing content trends over time and monitoring deep websites to provide statistical tracking reports for changes [38]. Estimating aggregates over a deep web repository like average document length, repository size estimation, generating content summary and approximate query processing [144] are also possible with deep web data at hand. Meta-search engines, price prediction, shopping website comparison, consumer behavior modeling, market penetration analysis and social page

evaluation are a number of example applications that accessing deep web data can enable [144].

2.3 STATE-OF-THE-ART IN ACCESSING DEEP WEB

This section provides a general overview on the suggested approaches for providing access to data residing in deep web sources.

2.3.1 Giving indexing permission

The most primitive solution to access data in a deep website is to get permission to access a full index of that website. Some data providers allow product search services to access and index the data available in their databases. However, this approach is not applicable in a competitive and uncooperative environment where the owners of a website are reluctant to provide any information that can be used by their competitors. For instance, information about size, ranking and indexing algorithms, underlying database features and the residing data in databases are denied to be accessed.

2.3.2 Harvesting all data

In this approach, all the available data in a deep website that is of interest of a user is extracted. The user information needs are answered by posing queries on this extracted data [14,25,71,105,117,139]. This method enables the centralized searching of web data sources. It uses a website's search form as an entry point to extract data in that website. Having filled in the input fields of the form, content of resulting pages are retrieved automatically. Having stored the extracted data, it is possible to pose user queries.

In order to get all data in a website, a number of challenges, such as smart form filling, structured data extraction and automation, scalability and efficiency, should be addressed [6,14,25,56,71,105,105].

2.3.3 Virtual integration of search engines

This method tries to automatically understand forms of different websites and provide a matching mechanism which enables having one mediated form [46, 56, 68, 69, 103–105, 116, 117, 132, 139]. This mediated form sits on top of the other forms and is considered as the only entry point for users. The submitted

queries to this mediated form are translated into queries that are acceptable by forms of the other deep web repositories. In this process, techniques like query mapping and schema matching are applied.

As the systems based on this technique need to understand semantics of provided entry points of deep web repositories, it requires more effort and time to apply this technique to more than one related domain. Difficult tasks of defining boundaries of domains on the web data and identifying which queries are related to each domain make the costs of building mediator forms and mappings high [105]. Two example systems, following this method, are described below.

Example 1: *MetaQuerier* Chang *et al.* [34] suggest a system based on having mediated schema. The proposed system abstracts from the forms of web databases through providing a mediated schema [139]. They limit the studies to one domain. In this selected domain, deep web sources are collected and their query capabilities are extracted from their interfaces. This information is used to cluster interfaces into subject domains. The front-end of MetaQuerier uses the discovered semantic matchings from deep web repositories to interact with users in the form of a domain category hierarchy.

Example 2: *Integraweb* Osuna *et al.* [117] suggest a system named *Integraweb* which issues structured queries in high-level languages such as SQL, XQuery or SPARQL. The authors claim that this usage of high-level structured queries leads to integrate deep web data with fewer costs than using mediated schemes through abstracting away from web forms. In virtual integration approaches, the unified search form abstracts away from the actual applications. Using structured queries over these mediated forms helps to have a higher level of abstraction [117].

As another method of accessing data without the need to fully harvest sources, there are approaches in the literature that suggest to query sources with suitable query plans and combine the resulted answers from different sources [16, 26, 27, 96].

2.3.4 Surfacing the deep web

Madhavan *et al.* [105] suggest to get enough appropriate samples from each interesting deep web repository so that a deep website has its right place among returned results by a search engine for a given query. This is done by precomputing the most relevant submissions for HTML forms as the entry points

to those deep web repositories. Then, the off-line generated URLs from these submissions are indexed and added to the indices of the surface web. The rest of the work is performed as if it is a page in surface web. The system presents search results and snippets to users. By clicking on any of the results, users are redirected to the underlying deep website, retrieving the fresh content [105].

2.3.5 Focused web harvesting

In "surfacing the deep web" [105], the goal is to sample a deep website so that it is well-presented in a general search engine to be matched against submitted queries. Surfacing approaches try to cover all the topics in a website. However, in focused web harvesting [84, 86], harvesters focus on extracting all relevant information to a given query, topic or entity.

Against focused crawlers that attempt to download pages that are similar to each other by predicting the probability of a page being relevant before downloading it [6, 32, 89, 106, 110], in focused web harvesting, the content is already indexed and searchable (e.g. in general search engines such as Google and Bing). Therefore, in focused web harvesting approaches, the content of pages can be extensively used and matched against queries to examine the relevance of pages to given query topics. Our previous work [84, 86] contributes mostly to this deep web access category.

2.4 Challenges in accessing deep web data

In accessing deep web data, a number of decisions can completely change the nature of challenges which need to be addressed. The most effective issue in defining requirements for a deep web access approach is the goal of that access.

1. If the reason behind accessing deep web data is answering queries over a limited number of domains then, the mediated form is the best access method. This will change the challenges to be addressed.
2. If the reason is improving the position of a deep website among the returned results from a search engine then, it is enough to have a number of distinct samples covering all the different aspects of a deep website.
3. If the goal is to keep track of changes in data from deep web sources or to provide statistics over it then, complete extraction and storage of data from deep web sources are necessary.

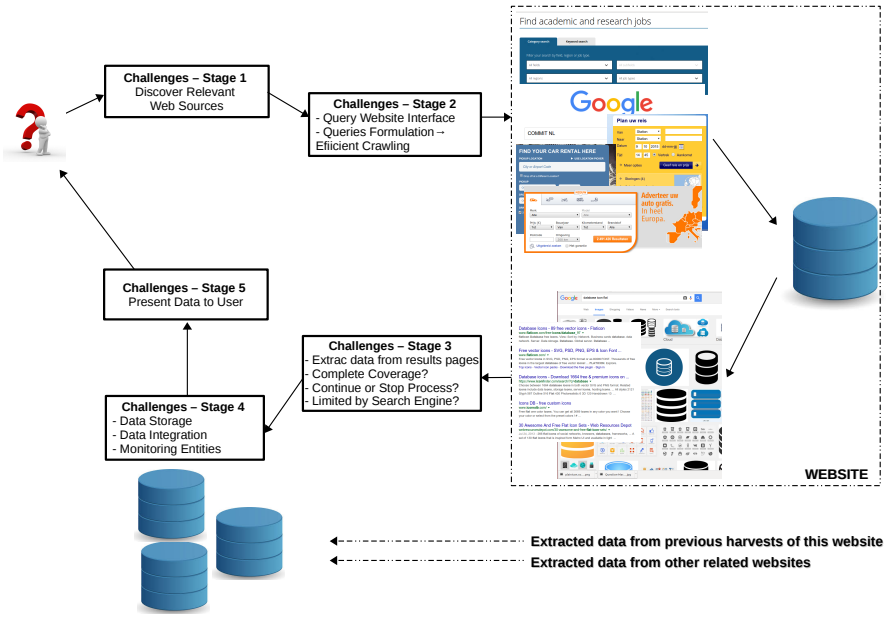


Figure 2.2: Challenges in the web data access cycle [144]

In addition to the access goal, the type of a deep website can affect the faced challenges. For example, the requirements to target a private or public source are different.

To have a harvester that automatically finds websites of interest, queries them, navigates through search results, extracts data, aggregates all information, filters out noise and presents to users what they asked for, the challenges mentioned in the following sections should be resolved. These challenges are also illustrated in the cycle of web data access for a user in Figure 2.2.

2.4.1 Deep web source discovery

To answer the information needs of a user, it is necessary to know from which data sources that information can be obtained. In the surface web, general search engines use indexes and matching algorithms to locate those sources of interest. In deep web sources, the data is hidden behind web search forms and

far from the reach of search engines. To discover which deep web data sources contain the data of interest, the following questions should be answered [144]. These are the challenges faced in `Challenges - stage 1` depicted in Figure 2.2.

1. Considering the huge number of websites available on the web, how can one determine the potential deep web sources for answering a given query?
2. How can we decide if a URL is a deep web repository? To answer this question, it is necessary to find forms in a website and decide if the forms belong to a deep website or not.
3. How can one determine if a URL is of a given topic and related to a given query?

2.4.2 Access data behind web forms

Having discovered the related deep websites to a given query, the data to answer the query should be retrieved. As mentioned earlier in this section, there are a number of different ways to access deep web data. As our goal is to harvest all the data in a deep website, the challenges depicted in Figure 2.2 should be addressed.

- How can web forms be efficiently and automatically filled [144]? Which form input fields should be filled? What are the bindings and correlations among inputs?

  For instance, in querying a website, detecting the interface, its type and recognizing different features of web forms can help harvesters. In a form-like query interface, the input fields that must be filled simultaneously to be able to return results should be determined. In addition, the fields whose values depend on each other such as minimum and maximum fields should be detected. Understanding the indexing and search policies of websites can also help to address the challenges in this stage. For further information on types of query interfaces and indexing and search policies, you can refer to Section 6.4.2.

  It is also important to study what values to submit to the input fields so that a harvesting process is more efficient with fewer queries but more results and fewer empty and duplicated pages. These questions are presented in `Challenges - stage 2` in Figure 2.2.

- How can a harvester navigate through returned results and extract data or entities from them [144]?

  Facing a long list of returned results for a submitted query, it is important to have tools which can go through all the returned results automatically and extract the required information. To do so, the following questions should be answered. How are returned results presented? How is the data presented in returned pages? Which data should be extracted and where is it placed in the page? Detecting data layout, data type, content format, language and navigation policies is a challenging task but necessary for extracting information from deep web sources. For further information and examples, you can refer to Section 6.4.3.

  After each page extraction or rather a new request to a website, the following questions should be answered. What is the status of a harvesting process; stop or continue? What is the size of the targeted deep web source and what percentage of that website is harvested? Is it possible to send a new request or a harvester is limited by the targeted search engine? How can a harvester cover all the data in a source considering limitations on the number of query submissions and returned results? How can a harvester keep the costs of a harvesting process low? How can extracted results from deep web sources be used as feedback to improve a harvesting process as it continues?

  It is also important to detect empty and duplicated pages and repeated information. These are the targeted questions in `Challenges - stage 3` in Figure 2.2.

- How should a harvester store the extracted data? How should a harvester perform entity identification, entity deduplication and detecting relations among entities to have a high-quality information extraction?

  As mentioned earlier in Section 2.2, the deep web data has high quality as it is mostly structured and resides in domain specific databases. How can we keep this quality after extracting data from websites? How can entity identification and detecting relations among entities help to improve a harvesting process as it proceeds? These are the questions faced in `Challenges - stage 4` shown in Figure 2.2.

- How can a harvester monitor changes of data and entities over deep web sources?

To monitor changes of data, firstly, we need approaches to detect changes. How can we develop efficient methods to detect changes of data/entities in one or several deep web sources? Let's assume that an entity is found in several deep web sources, how should the change in one website be treated and interpreted and which version should be judged as being reliable? These are questions that need to be addressed in `Challenges - stage 4` depicted in Figure 2.2.

- How should the extracted, stored and refined data be presented to users?

  What information should be the output of a web harvesting system; the entities and their relations, all the changes of an entity over time or the content of pages? How should this information be presented to users so that they can understand it, explore it and give feedback? These questions should be addressed in `Challenges - stage 5` shown in Figure 2.2.

2.5 TARGETED CHALLENGES

In this thesis, we target four main challenges. The focus of this thesis is on the challenges mentioned in `Challenges - stage 2`, `Challenges - stage 3` and `Challenges - stage 4` depicted in Figure 2.2.

First, we try to make deep web access approaches more efficient. To increase the efficiency of deep web access approaches, we study methods to estimate the size of non-cooperative data collections. This is one of the discussed issues in `Challenges - stage 3` depicted in Figure 2.2. To address this challenge, Chapter 3 reviews and categorizes the suggested approaches in the literature. Approaches from each category are implemented and compared in a real environment. Finally, four methods based on the modification of the available techniques are introduced and evaluated.

In the second targeted challenge, we study ways to improve data coverage for a given query in a harvesting task. In this study, we also investigate the application of feedback data for more efficient query generation mechanisms resulting in improved performance of a harvester. These issues are discussed in `Challenges - stage 3` and `Challenges - stage 4` that are depicted in Figure 2.2. To address this challenge, Chapter 4 proposes a new approach that automatically collects the related information 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 analyzing the retrieved

results and combining this analyzed information with information from a large external corpus.

The third challenge in this thesis is dedicated to investigate methods for detecting data changes on the web for a given query over time. How can we harvest only the changed data? How can we keep the harvested data up-to-date? How can a harvester monitor the changes of pages of interest? What is the most efficient way of detecting changes in a web data repository? To answer these questions, Chapter 5 introduces a new approach to efficiently harvest all the new and changed documents matching a given entity by querying a web search engine. This approach performs based on analyzing the content of retrieved results and a list of words selected from changed documents.

As the last challenge, in Chapter 6, we investigate all the important features in designing and developing an access approach that can be applied to a wide range of websites, domains and tasks. We also study if these features can form a basis for a performance comparison metric for harvesters.

CHAPTER 3

SIZE ESTIMATION OF NON-COOPERATIVE DATA COLLECTIONS

Is it possible to estimate the size of a website only through its query interface without any extra information? What are the state-of-the-art approaches and how we can improve them?

The content of this chapter is based on [80, 81].

Accessing data in a website can be a costly and time-consuming process. The knowledge of a data source size can enable web access methods to make accurate decisions on when to stop harvesting or sampling processes to avoid unnecessary query submissions [101]. This tendency to know the size of a data source is increased by competition among businesses on the web in which the data coverage is critical. This information is also helpful in the context of quality assessment of search engines [23, 133], search engine selection in federated search and resource/collection selection in distributed search [140]. In addition, it gives an insight over some useful statistics for public sectors like governments. In any of the above mentioned scenarios, when facing a non-cooperative collection which does not publish its information, the size of a collection has to be estimated [127]. This chapter reviews and categorizes suggested methods in the literature. Approaches from each of the categories are implemented and compared in a real environment. Finally, four methods

based on the modification of existing techniques are introduced and evaluated. In our suggested solution, the estimations from other approaches are improved ranging from 35 to 65 percent.

3.1 INTRODUCTION

With the increasing amount of high-quality structured data on the web, accessing data in deep web sources has gained more attention. As harvesting or sampling processes for these sources tend to be costly, with regards to the number of submitted requests, it is important to devise methods which can improve the efficiency of these processes. As discussed also in Chapter 1 and 2, if these access approaches can make accurate decisions about the best time to stop a harvesting process, they can already reduce the number of requests. The knowledge of a data source size can enable these algorithms to make accurate decisions on whether to continue or stop a harvesting process as the process proceeds [101].

The tendency to know the size of a website is increased by competition among businesses on the web (i.e. jobs and state agencies) to assure their customers of receiving the best possible services [41]. Also, in the context of search engines, the size can highly affect a search engine's quality assessment [23]. In addition, in federated search engines, the information about a website's size is helpful for selection of search engines to satisfy the information needs of a posed query. This is also useful in resource selection in distributed search domain [140]. In addition to these advantages, knowing about the size of a data collection can give an insight over some useful statistics for public sectors and governments. For example, knowing about the sizes of job offering websites can help to monitor job growth in a society [41].

In any of the above mentioned scenarios, when facing a non-cooperative collection that does not publish its information, the size of the collection has to be estimated [127]. In most of the cases, even if the size is published, it can not be trusted due to keeping information from competitors. As the only way of accessing these collections is through their query interfaces, the estimating methods should be able to perform by using only a query interface. In addition, the methods should be able to provide accurate estimations and be applied to any sets of documents [23].

Since 1998 that the problem of size estimation of non-cooperative data collections was introduced by Bharat *et al.* [19], several techniques to find new solutions have been proposed [7, 8, 11, 12, 23, 28, 41, 99, 102, 127, 134, 140]. These

techniques are divided into two main categories; relative size and absolute size estimators. Relative size estimators provide information on the size of a data collection relative to sizes of other collections while absolute size estimators estimate the absolute size of a collection. From the first category, the suggested methods by Bharat *et al.* [19] and Gulli *et al.* [63] are selected and discussed in this chapter. Approaches introduced in the second category are further classified based on a number of different technical aspects described below.

Using content or IDs of documents We can divide the size estimation of approaches based on what information from the returned results they use as the input to their algorithms. In the first category, there are approaches that analyze the content of selected documents in creating samples (e.g. pool-based approaches like Broder *et al.* [23]). The second category includes approaches that need to know only about the IDs of returned documents. Sample Resample, Capture History, Multiple Capture Recapture (MCR) [127] are examples for this category.

How to deal with bias The introduced approaches for collection size estimation follow *Query-Based Sampling* (QBS) method [19, 99]. In QBS, by sending a query to a search engine, the returned set of documents is considered as a sample. In this approach, it is assumed that samples are generated randomly, while in reality, chosen query, content of a document, ranking mechanism and many other factors affect the probability of a document to be selected. This makes the selection process not random and introduces biases in estimations. Based on the type of applied methods to deal with these biases, size estimation methods are further classified into the following sub-categories.

1. Approaches that try to reduce bias by using techniques to simulate random sampling to get close to a set of randomly generated samples (e.g. Bar-Yossef *et al.* approach [11, 12]). They can also apply techniques to prevent and remove bias.

2. A number of estimation methods accept the non-randomness of generated samples and try to remove the known biases (e.g. Heterogeneous-Ranked Model [102], Multiple Capture-Recapture Regression [127], Capture History Regression [127] and Heterogeneous Capture [140]).

3. This category includes approaches that accept samples as they are and do not try to remove the possible biases. These methods have high potential

to produce biases in estimations (e.g. Sample Resample, Capture History, Multiple Capture Recapture [127] and Generalized Multiple Capture Recapture [131]).

A general overview on the mentioned issues in this section is illustrated in Figure 3.1.

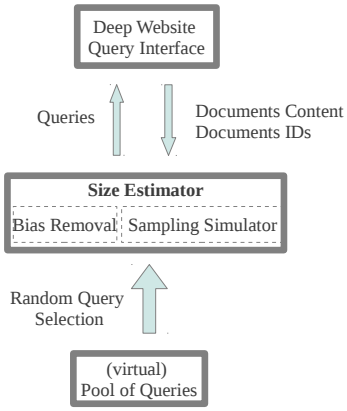


Figure 3.1: The general overview on data collection size estimators

Contributions The first contribution of this chapter is an experimental comparison among a number of size estimation techniques. Having applied these size estimation techniques on a number of real search engines, it is shown which technique can provide more promising results and what are the problems and shortcomings. As the second contribution, in addition to this experimental study, a number of modifications to the existing approaches are suggested. The extents to which these modifications improve the size estimations are also calculated and presented.

Outlook In Section 3.2, methods from each one of the three mentioned categories (classified based on techniques of dealing with biases) are introduced and discussed. The experiments on these approaches are explained in Section 3.3. In Section 3.4, improvements to the implemented estimation methods are discussed and tested. The results of these experiments and the analysis of these results are also presented in Section 3.4. Finally, the conclusion and future work are discussed in Section 3.5.

Table 3.1: Notations

Notation	Meaning
N	Absolute size of a collection
$\overline{N}$	Estimated size of a collection
A and B	Pools of queries
$\|A\|$	Number of queries in pool A
D	Collection of documents
D_A	Documents represented by queries in pool A
N_{D_A}	Number of documents represented by queries in pool A

3.2 BACKGROUND

Data collection size estimation approaches root in applied techniques for estimating human or animals population and earlier applications for fish and duck populations [7]. Pierre Laplace (1749–1827) estimated the human population of France through Equation 3.1. These methods are based on the ratio between the known (marked) and unknown (unmarked) parts of a collection. As mentioned before, these approaches are classified into two absolute and relative size estimators. The relative size estimators are further divided into three classes based on their applied techniques to deal with biases. In the following sections, sample approaches from each of these three classes are described. The notations used in this chapter are listed in Table 3.1.

$$\overline{N_{\texttt{France}}} = \#\texttt{PeopleInSampledCommunities} \times \frac{\#\texttt{AnnualBirth}}{\#\texttt{AnnualBirthInSamples}} \tag{3.1}$$

3.2.1 Approaches accepting samples as-they-are and no bias removal

Sample resample approach In *Sample Resample (SRS)*, the initial query is selected from a list of terms [28, 127]. This query is posed to a search engine and its returned results are considered as members of the first sample. The next queries are randomly selected from one of the returned documents by the previously submitted queries. This sampling process stops after downloading a predefined number of documents. With document frequency of a term in the

sampled documents and its frequency in the collection, the size of the collection is estimated. If the document frequency of a particular term t in a sample of m documents is df_t and its document frequency in the collection is Df_t, then, the collection size is estimated through Equation 3.2 [127].

$$\overline{N} = m \times \frac{\mathtt{Df}_t}{\mathtt{df}_t} \tag{3.2}$$

Capture recapture approach *Capture Recapture* method roots from ecology and is based on the number of duplicates among different captured samples. For example, to estimate the size of a type of animal like tigers first, a number of tigers are captured, marked and released. After a while, tigers are captured again. By counting the number of marked tigers in the second capture, the number of duplicates in these two samples is determined. Then, by applying Equation 3.3, it is possible to estimate the number of tigers [7] assuming that tigers are captured at random.

The use of this technique in data collection size estimation was first introduced by Liu *et al.* [99]. However, Liu *et al.* did not describe how to implement the proposed approach in practice. In their work, it is unclear what the sample size should be and how a random sample is chosen from a non-cooperative collection.

When applying Equation 3.3 in practice, if two samples are not big enough to have any duplicates, it is impossible to have any result. As a solution, multiple and weighted capture recapture methods are introduced. These techniques are explained in the following sections.

$$E(\overline{N}) = \frac{|\mathtt{FirstSample}| \times |\mathtt{SecondSample}|}{|\mathtt{DuplicatesAmongSamples}|} \tag{3.3}$$

Multiple capture recapture (MCR) To resolve the previously mentioned issue resulted from no duplicates among captured samples, Shokouhi *et al.* introduced a weighted method [127]. In this method, called *Multiple Capture Recapture*, by gathering T random samples of size m and counting duplicates within each pair of samples, the expected size of a collection is calculated through Equation 3.4. This approach performs based on the identifiers of documents to estimate the size of a collection. In this equation, $\frac{T\times(T-1)}{2}$ is the total number of pairs of samples. If there is an equal probability for each document to be selected, a document has a chance of $\frac{m}{N}$ to be in a sample and consequently $\frac{m^2}{N^2}$ chance of being in two samples. As there are N documents, there-

fore, the expected number of duplicates is calculated by $E(\texttt{duplicates}) = N \times \frac{T \times (T-1) \times m^2}{2 \times N^2}$ [131]. In case of observing dup number of duplicates for a pair of samples p, the size of the collection can be estimated by Equation 3.4.

$$\overline{N} = \frac{T \times (T-1) \times m^2}{2 \times \sum_{i=1}^{\#\texttt{AllPairs}} \texttt{dup}_{p_i}} \tag{3.4}$$

Generalized multiple capture recapture (GMCR) The MCR method can be applied only to samples of the same size. However, it is difficult to obtain samples of a uniform size. This restricts the use of MCR. Thomas [131] suggests a generalization over the MCR approach that allows its application on samples of different sizes through Equation 3.5. In this equation, m_x and m_y represent sizes of samples x and y respectively and $dup_{x,y}$ is the number of duplicates in these two samples. This approach is called *Generalized Multiple Capture Recapture (GMCR)*

$$\overline{N} = \sum_{x=1}^{T-1} \sum_{y=x+1}^{T} \frac{m_x m_y}{\texttt{dup}_{x,y}} \tag{3.5}$$

Capture history (CH) Shokouhi *et al.* suggest a weighting function for the capture recapture technique [127]. This approach is called *Capture History (CH)*. The CH approach estimates the size of a collection through Equation 3.6, by using the total number of documents in a sample (m_i), the number of documents in the samples that were already marked (MD_i) and the number of marked documents gathered prior to the most recent sample ($TotalMD_i$). In CH, it is assumed that the probability distribution of each individual satisfies a uniform distribution. However, this is not the case in search engines and hence, it causes bias in estimations.

$$\overline{N} = \frac{\sum m_i \texttt{TotalMD}_i^2}{\sum m_i \texttt{MD}_i} \tag{3.6}$$

Broder *et al.* - extra pool In "Estimating Corpus Size via Queries" [23], two approaches are introduced based on a basic estimator. In this basic estimator, both documents and queries are assigned with weights. The weight of a document is defined as the inverse of the number of terms in that document which are also in a pool of queries ($\frac{1}{|\texttt{Terms}_d \subset \texttt{Queries}_{\texttt{pool}}|}$). The pool is pre-computed and

includes queries that can be uniformly sampled. Accordingly, the weight of a query is defined as the sum of the weights of all documents that contain that query. By calculating the average of query weights for n number of queries selected from pool A, through Equation 3.7, an approximation to the basic estimator $W_{A,D}$ is obtained.

$$W_{A,D} = E(\overline{X}) = \frac{\sum_{i=1}^{n} \texttt{QueryWeight}_i}{n} \tag{3.7}$$

The first introduced approach by Broder *et al.* [23] belongs to the third category and is described later. In the second method, two query pools A and B covering two independent subsets D_A and D_B of the corpus D are required. In this context, independence means that D_A and D_B may share documents but fraction of documents that belong to D_A should be the same whether we consider the entire corpus or just D_B [23]. This approach estimates only the part of the corpus in which the pools are uncorrelated. In practice, it might be hard to obtain such sets of queries. Equation 3.8 shows how to estimate the size of a collection in *Broder* et al. - *extra pool* method.

$$\begin{aligned}
\overline{N_{D_A}} &= |A| \times W_{A,D} \\
\overline{N_{D_B}} &= |B| \times W_{B,D} \\
\overline{N_{D_A \cap D_B}} &= |A \cap B| \times W_{A \cap B,D} \\
\overline{N} &= \frac{\overline{N_{D_A}} \times \overline{N_{D_B}}}{\overline{N_{D_{A \cap B}}}}
\end{aligned} \tag{3.8}$$

3.2.2 Approaches based on removing bias

In QBS, different factors like the chosen query, document properties and search engine's specifications affect a sampling process. Detecting all these factors and resolving them can be costly or even not possible in some cases [12, 102]. Therefore, some approaches focus on removing the generated biases by these factors.

Bharat *et al.* introduce two major biases called query bias and ranking bias [19]. The query bias addresses different chances of documents to be chosen for different queries. The ranking bias results from returning only the top-k results and the applied ranking algorithms in search engines causing bias in size estimations.

Regression equations Regression analysis is a statistical tool used for estimating a variable that is dependent upon a number of independent variables [77]. The regression analysis investigates the relations between these variables and also provides the degree of confidence that the prediction is close to the actual value. In regression analysis, variation in dependent variable is represented by a value shown as R^2. R^2 value (between zero and one) shows to what extent the total variation of the variable is explained by the regression. A high value of R^2 suggests the regression model explains the variation successfully.

In regression analysis, the omitted variables and closely-correlated independent variables (if their effects are difficult to separate) can create difficulties in an estimation process [77]. As mentioned in Section 3.2.1, MCR and CH approaches introduce biases in estimations that lead to underestimating a collection size. To compensate for the selection bias in these approaches, the relation between the estimated and actual sizes of a collection is approximated by regression equations. Shokouhi *et al.* [127] applies this idea on a number of training collections and proposes Equations 3.9 and 3.10. These approaches are called *MCR-Regression* and *CH-Regression*. In these equations, R^2 values indicate how well the regression fits the data collection [127].

`MCR-Regression:`

$$log(\overline{N_{\texttt{MCR}}}) = 0.5911 \times log(\overline{N_{\texttt{MCR-Regression}}}) + 1.5767 \qquad R^2 = 0.8226 \quad (3.9)$$

`CH-Regression:`

$$log(\overline{N_{\texttt{CH}}}) = 0.6429 \times log(\overline{N_{\texttt{CH-Regression}}}) + 1.4208 \qquad R^2 = 0.9428 \quad (3.10)$$

Xu *et al.* [140] suggest another approach called *Heterogeneous Capture*. In this method, capture probabilities of documents in a sampling process are modeled with logistic regression. When calculating these probabilities, the document and query characteristics are modeled as a linear logistic model through applying Equation 3.11. To apply this approach, k random queries are posed on a search engine and their returned results are captured and recorded. Through applying Equation 3.11 on these sets of captured documents for each query, the collection size is estimated.

$$\overline{N} = \sum_{d=1}^{\texttt{TotalMD}} \frac{1}{p_d}$$
$$p_d = 1 - \prod_{q=1}^{n}(1 - p_{dq})$$
$$p_{dq} = \frac{exp(\beta_0 + \beta_1.len_d + \beta_2.rank_d + \beta_3.tf_{dq})}{1 + exp(\beta_0 + \beta_1.len_d + \beta_2.rank_d + \beta_3.tf_{dq})} \tag{3.11}$$

In Equation 3.11, $TotalMD$ is the number of captured documents, p_{dq} is the probability of a document d being captured on q_{th} try, len_0 is the length of d, $rank_d$ is the static rank of d (estimated by the average place of d among all retrieved results from all queries), tf_{dq} is the frequency of q_{th} query in d, β_0, β_1, β_2 and β_3 are unknown parameters and n is the number of sent queries.

Heterogeneous ranked model (Mhr) Lu [101] introduces a model to reduce the ranking bias based on a previous work in which Lu *et al.* [102] try to remove the query bias by suggesting an equation between the overlapping rate and the percentage of examined data with the assumption of having random samples from a uniform distribution. In this equation, the overlapping rate is defined as the total number of all documents divided by the number of distinct documents cached during the sampling procedure.

By relating this overlapping rate to the capture probability of a document in any of sampling iterations and applying linear regression, through Equation 3.12, the *Heterogeneous Model (Mh)* method can estimate the size of a collection. In this formula, `TotalMD` is the number of distinct documents, $\overline{N}$ is the estimated collection size, `OR` represents the overlapping rate, `PR` is the percentage of documents from the collection and α is a factor affecting the relation between `OR` and `PR`.

$$\overline{N} = \frac{\texttt{TotalMD}}{\texttt{PR}} = \frac{\texttt{TotalMD}}{1 - \texttt{OR}^{\alpha}} \tag{3.12}$$

In randomly selected documents from a uniform distribution, α is set to -2.1. In the absence of a uniform distribution, α is calculated through cv that determines the degree of heterogeneity for the distribution of capture probabilities of documents. The value of `cv` is estimated based on the history of captures through Equation 3.13. In this equation, f_i is the number of documents captured i times. T is the total number of captures of documents and

$\overline{N}_1 = \frac{\texttt{TotalMD}}{1-\texttt{OR}^{-1.1}}$ is the initial collection size estimation. With an estimated $\texttt{cv}$, the α is calculated by $\alpha = \frac{-2.1}{1+(1.1786\times \texttt{cv}^2)}$ [102].

$$\texttt{cv}^2 = \overline{N}_1 \frac{\sum_{i=1}^{n} i \times (i-1) \times f_i}{T \times (T-1)} - 1 \tag{3.13}$$

Lu *et al.* [102] suggest that this method can resolve the query bias and can only be applicable to search engines without overflowing queries. The overflowing queries are the queries for which there are more matched results than returned ones. This problem is addressed by Lu [101].

Lu [101] suggests multiplying the model introduced in Equation 3.12 by overflowing rate of queries as shown in Equation 3.14. Overflowing rate ($\texttt{OF}$) is calculated by dividing the total number of matched documents for a query by the total number of returned documents for that query. This model is named as the *Heterogeneous-Ranked Model (Mhr)*. If the total number of returned results for a query (matched documents) and the number of results that user can view from the set of matched documents is not available, the model becomes similar to Mh model [102]. Formula 3.14 estimates the size of a collection through Mhr model. In this formula, $\texttt{TotalMD}$ represents the total number of distinct documents.

$$\overline{N} = \texttt{OF} \times \frac{\texttt{TotalMD}}{1-\texttt{OR}^{-1.1}} \tag{3.14}$$

3.2.3 Having close-to-random samples and bias removal

One of the methods to have random or close-to-random samples is to apply stochastic simulation techniques like Monte Carlo methods [59]. From Monte Carlo simulation methods, rejection sampling, importance sampling and metropolis-hastings are applied for size estimation in the literature [11,12].

Rejection sampling, importance sampling and metropolis-hastings methods are based on producing biased samples and weights for sampled documents representing their capture probabilities. The availability of these weights allows the application of stochastic simulation methods [12]. The stochastic simulation techniques accept the samples from a trial distribution $Q(x)$ and simulate sampling from a target distribution $P(x)$. Therefore, by defining a $Q(x)$ which has uniform distribution and can be easily sampled, the unbiased sampling is done for $P(x)$ [59]. Samples that are not in $P(x)$ are ignored.

In rejection sampling, it is assumed there is a $Q(x)$ with a predefined constant c that $P(x) < c \times Q(x)$. If generated samples from $Q(x)$ satisfy this inequality, they are considered to be also in $P(x)$ [59]. In importance sampling, instead of generating samples from a probability distribution, the focus is on estimating the expectation of a function under that distribution [59]. For each generated sample, a weight is also introduced. This weight is used to represent the importance of each sample in the estimator.

Broder *et al.* - sampling Broder *et al.* suggest two approaches [23]. In the sampling method, the size is estimated through Equation 3.15 using the size of a pool ($|A|$), a basic estimator $W_{A,D}$ and the ratio between the number of documents represented by queries in the pool and the collection size ($r_A = \frac{N_{D_A}}{N_D}$). As this estimation can be costly, the ratio is estimated by sampling documents.

To calculate $W_{A,D}$ through Equation 3.7, we need weights for queries and documents. The weight of a document is estimated by calculating the number of terms in that document that are also in the pool ($\frac{1}{|\texttt{Terms}_d \subset \texttt{Queries}_{\texttt{pool}}|}$). Accordingly, the weight of a query is defined as the sum of document weights of all documents that contain the query.

Calculating weights for queries implies that this approach is implicitly using the importance sampling [11]. In this method, it is not studied how the difference between the predicted and the actual document weights can cause bias [11].

$$\overline{N} = \frac{|A|}{r_A} \times W_{A,D} \tag{3.15}$$

Bar-Yossef *et al.* approach In Broder *et al.* method, the assigned weight to a document is predicted and might be different from the actual weight as there is not enough knowledge of parsing, indexing and search algorithms of a search engine and also the effect of having only top-k results. This difference between the actual and predicted weights is defined as degree mismatch [12].

To resolve the degree mismatch, Bar Yossef *et al.* suggest defining a sample space as a pair of query and document represented as (q, d) [11]. This sample space definition eliminates the use of rejection sampling for the random selection of queries [12]. Instead of sampling from a target distribution, the estimator samples a document from a different trial distribution that allows easier random sampling (i.e. importance sampling). The estimator considers the degree mismatch by defining a *valid query-document* graph. In this valid

graph, queries and documents are presented as nodes and there is an edge between a query and document if the document is returned for the query and also contains that query. By using this valid sample pair, the collection size is estimated through Equation 3.16.

$$\overline{N} = \frac{1}{\texttt{Times}} \sum_{i=1}^{\texttt{Times}} \texttt{PSE} \times \pi_D(d_i) \times \texttt{deg}_v(q) \times \texttt{IDE}(d_i) \tag{3.16}$$

In Equation 3.16,

- PSE is the size of a pool which contains only queries that are in the valid graph. PSE is estimated through random sampling.
- Times is the number of repetitions of an estimation process.
- $\pi_D(d)$ represents the weight of document d in a target measure on the set of documents indexed by search engine D. In a uniform target measure, $\pi_D(d) = 1$, and in non-uniform target measures, $\pi_D(d) = \texttt{length}(d)$ or $\pi_D(d) = \texttt{PageRank}(d)$.
- $deg_(q)$ is *query degree* and equals with the number of documents connected to query q in the valid graph.
- $\texttt{IDE}(d)$ is the estimator of the inverse of document degree ($\frac{1}{deg_{(d)}}$).

The document degree is calculated through either (1) a brute force calculation which is precise but costly, (2) ignoring the search engine and using the pool which is cheap but not accurate, (3) a sampling method which is biased or (4) estimation of $\frac{1}{deg_{(d)}}$ referred to as $\texttt{IDE}(d)$ in Equation 3.16.

By submitting a number of randomly selected queries to a search engine, Bar Yossef *et al.* method examines the results of each query to find a valid document-query pair. The procedure stops when it reaches a query that has at least one valid result. This pair is considered as a sample. Then, the document degree is calculated for this sampled document by querying the search engine by terms in that document that are also in the pool. If that page is among the query results, the inverse document degree estimation procedure stops. The number of sampled queries is considered as success parameter ($\texttt{IDE}(d) = \frac{1}{deg_d} = \frac{|\texttt{sampledQueries}|}{|\texttt{Queries}_{\texttt{pool}}(d)|}$).

With the knowledge of the number of documents in the valid graph for the sampled query, the estimation of the inverse of document degree and the size of the pool of valid queries (estimated through random sampling), the size of collection is estimated through Equation 3.16.

3.2.4 Overview of related work

Table 3.2 represents a summary of all the mentioned approaches from the literature. In this table, the formulas and categories of the discussed collection size estimation methods are mentioned.

Table 3.2: Overview of data collection size estimation methods

Approach	Document ID/Content	Formula
MCR **[127]**	ID	$\overline{N_{\texttt{MCR}}} = \frac{T \times (T-1) \times m^2}{2 \times \sum_{i=1}^{\#\texttt{AllPairs}} \texttt{dup}_{p_i}}$
MCR-Regression **[127]**	ID	$log(\overline{N_{\texttt{MCR}}}) = 0.5911 \times log(\overline{N_{\texttt{MCR-Regression}}}) + 1.5767,$ $R^2 = 0.8226$
CH **[127]**	ID	$\overline{N_{\texttt{CH}}} = \frac{\sum m_i \texttt{TotalMD}_i^2}{\sum m_i \texttt{MD}_i}$
CH-Regression **[127]**	ID	$log(\overline{N_{\texttt{CH}}}) =$ $0.6429 \times log(\overline{N_{\texttt{CH-Regression}}}) + 1.4208,$ $R^2 = 0.9428$
GMCR [131]	ID	$\overline{N_{\texttt{GMCR}}} = \sum_{x=1}^{T-1} \sum_{y=x+1}^{T} \frac{m_x m_y}{\texttt{dup}_{x,y}}$
Broder *et al.* Sampling **[23]**	Content	$\overline{N_{\texttt{Broder.Sampling}}} = \frac{\lvert A \rvert}{p_A} \times W_{A,D},$ $W_{A,D} = E(\overline{X}) = \frac{\sum_{i=1}^{T} \texttt{QueryWeight}_i}{T}$
Broder *et al.* Extra Pool **[23]**	Content	$\overline{N_{\texttt{Broder.pool}}} = \frac{\overline{N_{D_A}} \times \overline{N_{D_B}}}{\overline{N_{D_{A,B}}}},$ $\overline{N_{D_A}} = \lvert A \rvert \times W_{A,D}$
Bar-Yossef *et al.* **[13]**	Content	$\overline{N_{\texttt{Bar-Yossef}}} =$ $\frac{1}{\texttt{Times}} \sum_{i=1}^{\texttt{Times}} \texttt{PSE} \times \pi_D(d_i) \times deg_v(q) \times \texttt{IDE}(d_i)$
HC **[140]**	Content	$\overline{N_{\texttt{HC}}} = \sum_{d=1}^{\texttt{TotalMD}} \frac{1}{p_d}, p_d =$ $1 - \prod_{q=1}^{T} (1 - p_{dq}) p_{dq} =$ $\frac{exp(\beta_0 + \beta_1 . len_d + \beta_2 . rank_d + \beta_3 . tf_{dq})}{1 + exp(\beta_0 + \beta_1 . len_d + \beta_2 . rank_d + \beta_3 . tf_{dq})}$
Mhr **[101]**	ID	$\overline{N_{\texttt{Mhr}}} = \texttt{OF} \times \frac{\texttt{TotalMD}}{1 - \texttt{OR}^{-1.1}}$

3.3 EXPERIMENTS

As one of the contributions of this chapter, an empirical study is performed on the suggested approaches for estimating the size of a collection. In this study, these approaches are applied to real data collections available on the web. We select websites that their sizes are known and represent different domains. In Table 3.3, a list of the applied websites in this experiment with their corresponding sizes is presented.

Table 3.3: Test set - real data collections on the web

Data collection	Size* (number of documents)
Personal website http://wwwhome.cs.utwente.nl/~{}hiemstra/	382
University search website http://www.searchuniversity.com/	4,076
Job search website http://www.monster.co.uk/	40,000**
Youtube education http://www.youtube.com/education/	311,00
English corpus of Wikipedia http://en.wikipedia.org/	3,930,041
US national library of medicine - English documents http://www.ncbi.nlm.nih.gov/pubmed/	17,606,509

* The sizes of collections are reported on 12/7/2012.
** Although the actual size is not published on the website, this is a close estimation calculated by browsing jobs in sections.

In our experiments, we attempted to implement techniques from all three sub-categories of the absolute size estimators category. Therefore, MCR, MCR-Regression, CH, CH-Regression, Mhr and Bar-Yossef *et al.* methods were chosen to be implemented.

Implementation differences In our experiments, if there are no found duplicates among samples, the number of duplicates is set to 1. This enables the approaches to provide an estimation even without any duplicate. In addition, for MCR, samples of a fixed size are required. Therefore, the average size of all samples is set as the sample size in calculations.

Performance measure To get a more accurate evaluation of an approach, each approach is repeated 100 times for a predefined sample size and number of sampling events. The results of these iterations are presented through calculating Relative Bias (RB) as shown in Equation 3.17 [102]. RB measures how close an estimation is to the actual size of a collection. In this formula, $E(\overline{N}) = \frac{\overline{N_1}+\overline{N_2}+...+\overline{N_{\texttt{Times}}}}{\texttt{Times}}$ represents the mean value of `Times` number of estimations.

$$\texttt{RB} = \frac{E(\overline{N}) - N}{N} \tag{3.17}$$

To provide a better comparison of performances of different approaches, in addition to `RB`, another measure is also used by calculating $Log10(E(\overline{N})/N)$. The results are shown in Figure 3.3. These two measures provide a more clear overview of performances and make comparisons easier.

3.3.1 Applied query pools

To apply the introduced techniques, three different query pools are developed: *pool A*, *pool B* and *pool C*. To create these pools, we try to follow the guidelines mentioned in the literature. Query pool A is created for Mhr, CH, MCR, G-MCR and the regression techniques. It includes the top 1000 most frequent words extracted from the pages of Wikipedia and the ClueWeb09 data set [120].

Query pool *B*, is created by following the mentioned guidelines by Bar Yossef *et al.* [13]. They propose two different pools for sampling. The first is for training purposes, a pool of 43 million phrase queries of length 4 extracted from pages in ODP data set [45]. However, for real cases on the web, pool *B* is created with 2.775 billion queries including 1.5 billion decimal strings of 5 to 9 digits, 7.4 million single terms extracted from Wikipedia, 18 million single terms extracted from the ClueWeb09 data set [120] and 1.25 billion two-term conjunctions of the 50,000 most frequent extracted single terms (excluding the 100 most frequent ones).

The third pool *C* is applied for M-Bar-Yossef approach that is introduced in Section 3.4.1. This pool consists of four different pools: *C-1*, *C-2*, *C-3* and *C-4*. These pools sequentially consist of the most 10^4, 10^5, 10^6 and 10^7 frequent terms extracted from the retrieved pages from Wikipedia and parts of the web (ClueWeb09 data set). In addition, for each pool, for the same amount of terms, integer digits are added doubling the initial size of each pool (2×10^4, 2×10^5, 2×10^6 and 2×10^7).

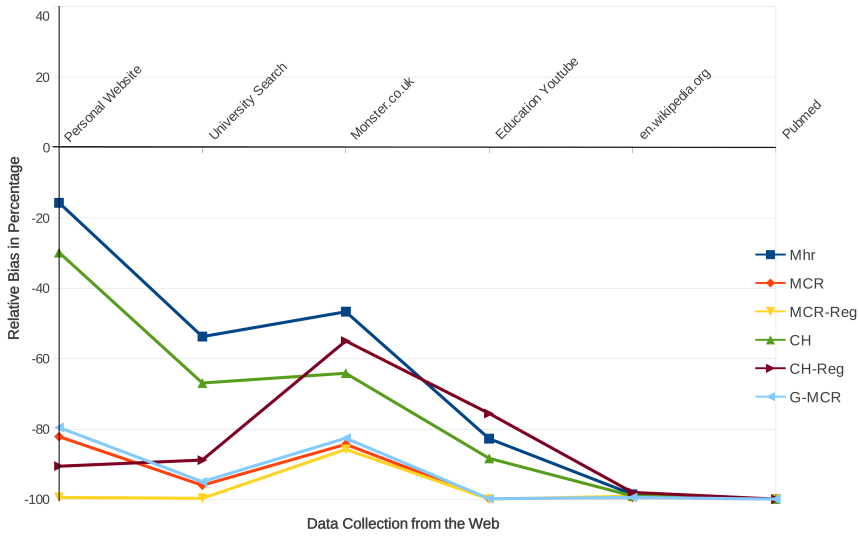


Figure 3.2: The average distance between the actual size of a collection on the web and its estimated size by the selected approaches from the literature in over 100 iterations (RB) (The lines are added only to provide more readability to the graph. The closer the points are to y=0, the more accurate estimations they represent.)

3.3.2 Results

Results of applying Mhr, MCR, G-MCR, CH, MCR-Regression, CH-Regression approaches on our test set are illustrated in Figure 3.2. These results are normalized by using the *Relative Bias* metric to compare performances of all the tested approaches on all the diverse data collections with different sizes. If the estimated size by an approach for a website is half of its actual size, the corresponding relative bias (RB) for that approach is -0.5 which relates to -50 percent in Figure 3.2.

It is important to note that estimations by Bar-Yossef *et al.* technique, using their suggested pool for real cases [13], failed in most of our experiments due to the excessive consumption of resources by the approach. As the application of Bar-Yossef *et al.* failed to return any result for our test cases, we do not include Bar-Yossef *et al.* in our results. However, to resolve this problem, a solution is suggested in Section 3.4.

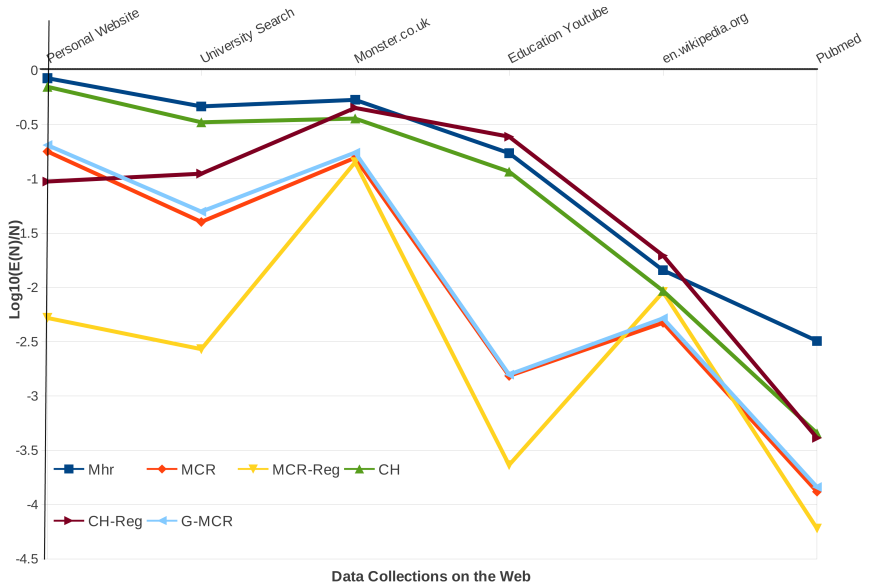


Figure 3.3: The log base 10 of the estimated size of a collection divided by its real size for all the selected approaches from the literature (The lines are added only to provide more readability to the graph. The closer the points are to y=0, the more accurate estimations they represent.)

3.4 Improvements

As our second contribution in this chapter, we attempted to improve the performance of the introduced size estimation approaches in Section 3.2. As a result, four different approaches are suggested based on modifying the presented approaches in the literature. These approaches are described in detail in the following sections.

3.4.1 Modified Bar-Yossef (M-Bar-Yossef)

The performance of Bar-Yossef *et al.* approach depends highly on the selection of a query pool. In our experiments, the application of a big pool, suggested by Bar-Yossef *et al.* to be used for real cases, resulted in such a costly process regarding query submissions that the experiment could not be continued. How-

ever, by adjusting the applied pools in our experiments, we observed that if a better pool is chosen, the results can surpass the other introduced approaches.

We define a better pool as a pool that covers more pages from a targeted document collection. To provide such pools for document collections with different features like size and domain, it is necessary to define different pools and select the best suitable pool among them for a given collection. In *Modified Bar-Yossef (M-Bar-Yossef)*, we suggest an intelligent pool selection to improve Bar-Yossef *et al.* technique. The M-Bar-Yossef can be applied for document collections with different sizes. In this method, the size and coverage of the applied pool are increased step by step and based on the obtained results from previously posed queries.

In M-Bar-Yossef, we suggest to start with a small pool. After sending a few queries, based on the number of found queries and documents in valid graph (described in Section 3.2.3), it is decided if a bigger pool suits this estimation or not. Then, if required, a bigger pool is selected and processed. The features of these pools are described in detail in Section 3.3. This pool selection process continues until the best pool or the biggest pool is reached. It is important to point out that the previously sent queries and found results are used in next steps of the algorithm and pools are already indexed. These issues make it possible to implement the intelligent pool selection without significant extra cost.

The performance of this approach is illustrated in Figure 3.4 (by RB) and Figure 3.5 (by $Log10(E(\overline{N})/N)$). Table 3.4 shows the performance differences between M-Bar-Yossef and other tested approaches when tested on websites in the test set (listed in Table 3.3). In Table 3.4, the average performances of approaches on all data collections are presented. As it is shown in this table, M-Bar-Yossef provides 35 to 65 percent closer estimations considering all the tested websites.

3.4.2 Modified multiple capture recapture (M-MCR)

We modify MCR based on the idea that diversity in samples provides more information for a size estimation process. To test this idea and improve the performance of MCR, similar samples are removed. The similarity between two samples is judged based on the number of duplicates between those samples. The similarity threshold should be adjusted which was set as 30 percent of sample size in this work. This modification is referred as *Modified Multiple Capture Recapture (M-MCR)*. The average performances of M-MCR in comparison to the other introduced approaches are shown in Table 3.4. In Figure 3.4

Table 3.4: Improvements resulting from the modifications

	Mhr	MCR	MCR-Reg	CH	CH-Reg	G-MCR
M-Bar-Yossef	36.25	63.67	67.36	44.74	54.70	62.77
M-MCR	-19.1	8.27	11.96	-10.6	-0.7	7.37
M-MCR-Reg	-24.1	3.25	6.94	-15.6	-5.7	2.34
M-CH-1	1.35	28.77	32.46	9.84	19.79	27.86
M-CH-1-Reg	2.50	29.92	33.60	10.98	20.94	29.01
M-CH-2	0.81	28.23	31.92	9.30	19.26	27.33
M-CH-2-Reg	2.77	30.19	33.87	11.25	21.21	29.28

Note: This table provides the resulted percentages of improvements by the modified approaches in comparison with the implemented approaches from the literature while considering the average results of a method for all the tested websites as its performance.

(RB) and Figure 3.5 ($Log10(E(\overline{N})/N)$), it is possible to compare M-MCR performance with all the other introduced approaches. This modification is also helpful to make a better decision on when to stop a sampling process.

The estimated size by this modified version of MCR is used by the regression formula introduced in MCR-Regression method, Formula 3.9. This is called *M-MCR-Regression* technique. The results and improvements of this approach are shown in Figure 3.4 (RB), Figure 3.5 ($Log10(E(\overline{N})/N)$) and Table 3.4.

3.4.3 Modified capture history (M-CH-1)

We suggest improving CH based on the same idea mentioned in M-MCR which is removing similar samples by counting duplicates between pairs of samples. If the number of duplicates between two samples is more than 30 percent of sample size, two samples are judged to be similar and only the earlier found sample is included in calculations. This also provides information on the best time to stop a sampling process. As there are two modifications introduced for CH, this modification is called as *Modified CH-1 (M-CH-1)*. The average performances of M-CH-1 in comparison to the other introduced approaches are provided in Table 3.4. In Figure 3.4, it is possible to compare the performance of M-CH-1 with all the other approaches. The improvements introduced by this approach are also presented in Table 3.4.

The estimated size by this modified version of CH method is used by the

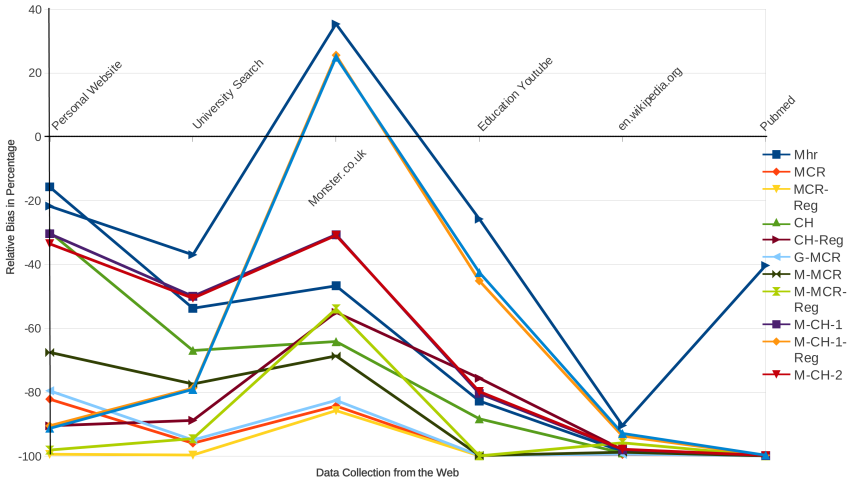


Figure 3.4: The average distance between the real size of a collection on the web and its estimated size by the implemented approaches from the literature and proposed approaches in this chapter in over 100 iterations (RB) are presented. The lines are added only to provide more readability to the graph. The closer the points are to y=0, the more accurate estimations they represent.

regression formula introduced in CH-Regression. This is referred as *M-CH-1-Regression* method. The results and improvements of this approach are shown in Figure 3.4 (RB), Figure 3.5 ($Log10(E(\overline{N})/N)$) and Table 3.4.

3.4.4 Modified capture history (M-CH-2)

As another method to improve the performance of CH, similar samples are judged based on the number of duplicates in a sample considering all the previously captured documents. If this number is more than 50 percent of a sample size, the sample is not included in calculations. This modification is referred to as *Modified CH-2 (M-CH-2)* technique. The results of its average performance over all the tested websites are shown in Table 3.4. In Figure 3.4, it is possible to compare the performance of M-CH-2 with all the other examined approaches.

The estimated size by M-CH-2 is used by CH-Regression equation and is called *M-CH-2-Regression*. The results and improvements of this approach are shown in Figure 3.4 (RB), Figure 3.5 ($Log10(E(\overline{N})/N)$) and Table 3.4.

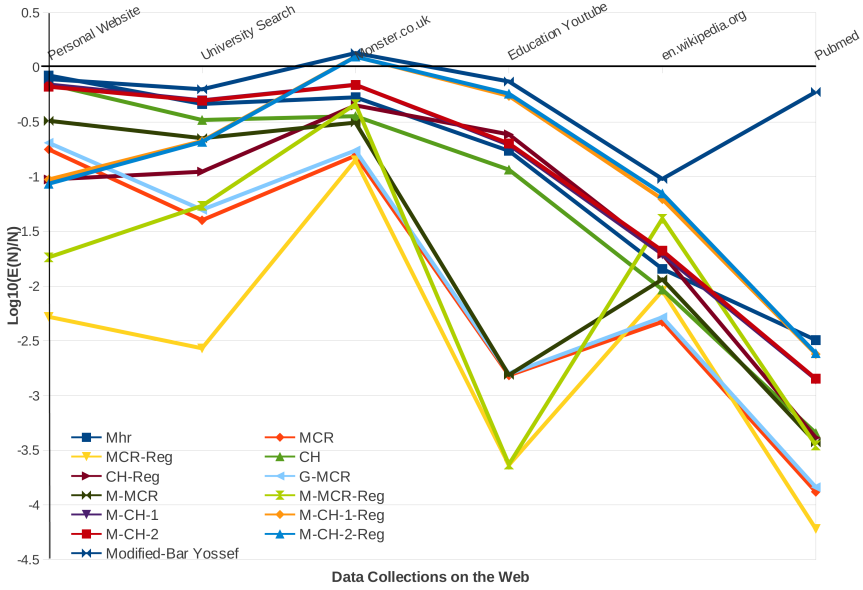


Figure 3.5: The log base 10 of the estimated size of a collection divided by its real size for each of the selected approaches from the literature and proposed approaches in this chapter are presented. The lines are added only to provide more readability to the graph. The closer the points are to y=0, the more accurate estimations they represent.

3.5 Conclusion and future work

In this chapter, we studied and compared the state-of-the-art approaches in the literature for estimating the size of a non-cooperative website. Multiple Capture Recapture, Capture History, Generalized Multiple Capture Recapture, Bar-Yossef *et al.* and regression-based approaches are selected to be studied and compared. To provide an appropriate comparison environment, we considered two issues. First, we defined our test set of websites on the web from different domains (such as job vacancies, wikis, articles and personal websites) with different sizes. The second issue was the information available for each approach. The number of sampling events and sample size were set to be the same for all of these approaches. This provided a fair comparison of performances. Although this test environment could be improved by adding more real websites, we believe that it provides enough test cases for comparing the

available size estimation approaches.

Among all the studied approaches, the modified version of Bar-Yossef *et al.* (M-Bar-Yossef *et al.*) provided 35 to 65 percent better size estimations for the tested websites on the web. However, M-Bar-Yossef *et al.* method can not be implemented for the websites which do not provide access to the content of search results. Therefore, if we need to estimate the size of a website for which you cannot access the content of search results, Mhr technique, both modified versions of CH method (M-CH-1 and M-CH-2) and their regressions (M-CH-1-Regression and M-CH-2-Regression) are among the options to be considered. These approaches had close estimations considering their average performances for all the tested websites.

This work, while providing a detailed overview on the available techniques and approaches applied for the size estimation of non-cooperative websites, still leaves a number of questions unanswered. One of these questions is: what is the most appropriate time to stop a sampling process? As one of the strategies, considering that all estimation approaches provided better results with more data, continuing as far as limitations permit is one of the options. The other alternative is to study questions like what is the adequate number of samples and the most appropriate sample size to provide the most accurate estimation.

As another future work, we can study more improvements to the suggested approaches in this work. As an example, in the selection of pools in M-Bar-Yossef *et al.* method, the selection procedure can be based on queries from different domains. This classification might lead to a higher accuracy in size estimations of domain-based document collections.

CHAPTER 4

TOWARDS COMPLETE COVERAGE IN FOCUSED WEB HARVESTING

Given the imposed limitations by search engines in both surface web and deep web and limited resources of harvesters, how can we improve the data coverage by a harvester? How can we achieve more coverage in harvesting all the related documents to a topic of interest, given these restrictions?

This chapter is based on [83, 84].

With the goal of harvesting all the information about a given entity, in this chapter, we try to harvest all the documents matching a given entity by querying a web search engine. The objective is to retrieve all the information about, for instance "Michael Jackson", "Islamic State" or "FC Barcelona", from the indexed data in search engines or hidden data behind web forms, using a minimum number of queries. Policies of web search engines do not usually allow accessing all the matching query search results of 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 that collects the information for a given query from a search engine with the mentioned limitations automatically. The approach minimizes the number of query submissions by

sending queries composed of the given query (*seed query*) and one carefully chosen additional word with the aim of retrieving a maximum number of new documents. The additional words are chosen by analyzing the retrieved results and combining this analyzed information with information from a large external corpus. The new approach outperforms the existing approaches when tested on Google, measuring the total number of unique documents found per query.

4.1 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 the relevant data to your topics 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 and biggest sources of data. The availability of an up-to-date crawl of the web can definitely facilitate collecting all relevant information to 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 provided search engines by such organizations.

Most web data is accessible by querying general search engines like Google or Bing 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 some of the research work in the literature [6, 14, 25, 71, 105], the focus is mainly on deep websites with form submissions. These studies investigate web forms, form fields' inputs, fields' bindings and other features of forms and websites that can influence harvesting a deep website and extracting information about a given entity. Instead, our work examines search systems providing keyword-based search interfaces[1]. This enables us to include any websites with keyword search.

With keyword search as the only way of accessing data, to achieve a full data coverage on a given query in a search system, the primitive solution that comes to mind is to follow these steps: 1) submit a query, 2) retrieve returned results from the results page, 3) go to the next results page and 4) repeat number 2 and 3 until there is no next page. For most queries with thousands of

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

results, going through these steps is a labor-intensive task. Web scrapers address this challenge by navigating through search results automatically and downloading the desired data. However, even with (semi-)automatic scrapers, achieving a 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 that a search engine allows a user to access is limited.

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

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

In this work, we rely on the potential power of search refinement techniques to uncover results beyond what a search engine allows a user to directly access due to $\#\texttt{ResultsLimited}$ and $\#\texttt{RequestsLimited}$ limitations. These techniques are typically based on adding extra terms to the initial query to obtain refined search results. We propose a method to refine search results for the purpose of achieving a full data coverage.

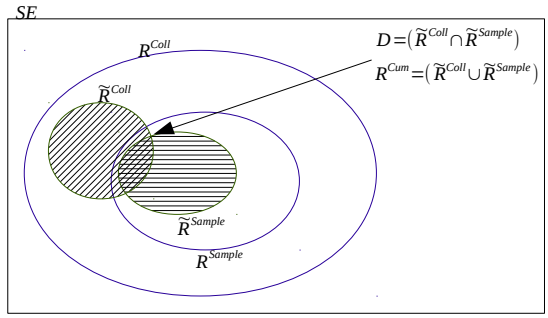
Problem definition For a given query $Q^{\texttt{Coll}}$, $R^{\texttt{Coll}}$ is the set of all documents containing $Q^{\texttt{Coll}}$. A search engine with $\#\texttt{ResultsLimited}$ prevents users from accessing all these documents. When submitting $Q^{\texttt{Coll}}$, users cannot view more documents than a predetermined number l. l can be different from the actual number of returned results ($\tilde{R}^{\texttt{Coll}}$) in case of underflowing queries[2]. For overflowing queries[3], we have $l < |R^{\texttt{Coll}}|$ leading to $\tilde{R}^{\texttt{Coll}} \subset R^{\texttt{Coll}}$. The number of documents returned for $Q^{\texttt{Coll}}$ ($|\tilde{R}^{\texttt{Coll}}|$) does not exceed l.

To retrieve all the members of $R^{\texttt{Coll}}$, we form a new query $Q^{\texttt{Sample}}$ by adding a term T to the $Q^{\texttt{Coll}}$ query ($Q^{\texttt{Sample}} = Q^{\texttt{Coll}} + T$). The documents matching this formed query is referred to as $R^{\texttt{Sample}}$ and the ones accessible by a user are defined by $\tilde{R}^{\texttt{Sample}}$. The present documents in both of $\tilde{R}^{\texttt{Sample}}$ and $\tilde{R}^{\texttt{Coll}}$ are defined as duplicates and referred to as D. We also define $R^{\texttt{Cum}}$ as

[2]An underflowing query is a query that produces fewer results than $\#\texttt{ResultsLimited}$ allows.

[3]An overflowing query is a query that produces more results than $\#\texttt{ResultsLimited}$ allows.

Figure 4.1: Problem definition



the union of all retrieved documents. Provided that $\tilde{R}^{\texttt{Sample}}$ does not include only duplicates, it provides new documents from $R^{\texttt{Coll}}$. Figure 4.1 models these definitions. To obtain all the documents in $R^{\texttt{Coll}}$, several iterations of forming new queries are performed. After each iteration of query reformulation and submission, $R^{\texttt{Cum}}$ and D are recalculated as shown in Equation 4.1. We continue these iterations till $R^{\texttt{Cum}}$ equals $R^{\texttt{Coll}}$. Now, the *main problem* to be addressed is how to minimize the number of these iterations to have an efficient method that complies with #RequestsLimited too.

$$R_n^{\texttt{Cum}} = \bigcup_{i \in [1,n]} \tilde{R}_i^{\texttt{Sample}}$$
$$D_i = \tilde{R}_i^{\texttt{Sample}} \cap R_{i-1}^{\texttt{Cum}} \tag{4.1}$$

In this approach, forming new 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 that return as many documents as l while minimizing $|D|$. This defines the goal of this work as follows:

Our goal is collecting the biggest possible set of documents that match a given query by posing the least possible number of requests to a search engine.

Maximizing $|\tilde{R}^{\texttt{Coll}}|$ while minimizing $|D|$ is a necessary requirement to achieve this goal. Minimizing duplicates becomes complicated with the presence of *ranking bias* and *query bias* [19]. The search engine's ranking algorithm (e.g. Google PageRank) and the selection of initial query favor some documents more than others to be returned by a search engine.

Therefore, we define the *main challenge* in this chapter as finding countermeasures for generated biases by ranking algorithms. These biases cause same documents to be always ranked high among returned results and therefore, be always present in $\tilde{R}^{\texttt{Coll}}$ and $\tilde{R}^{\texttt{Sample}}$s resulting a big D. To meet this challenge, several approaches are suggested, implemented and compared in this chapter. We test our approaches on Google that claims to have billions of URLs [62]. Google imposes both #`ResultsLimited` and #`RequestsLimited` and has ranking bias through using its PageRank algorithm.

The next section is dedicated to related work. Then, Section 4.3 discusses solutions classified into three main categories. In Section 4.4, these solutions are tested and the results are analyzed and presented. Finally, in Section 4.5, the drawn conclusions from these results are discussed and areas for further research are identified.

4.2 LITERATURE STUDY

Deep web harvesting In this work, we are interested in applied methods to access deep web data for either sampling or harvesting websites. Recent studies on accessing deep web data are mainly focused on websites requiring form submissions [6, 14, 25, 71, 105], studying a web forms to achieve an efficient access approach. For example, Madhavan *et al.* [105] investigate how to identify the values and types of different form fields and how to navigate the search space of possible form input combinations efficiently to avoid unnecessary submissions leading to a more efficient source sampling process. Although these studies do not specifically target focused web 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 some external domain knowledge.

Query-based sampling As discussed in Chapter 3, accessing data in all diverse websites that provide a search interface follows a similar method referred as *query based sampling (QBS)* (submitting requests to get results). In QBS, by sending a query to a search engine, the set of returned results is considered as a sample of documents [19, 28]. Having produced a number of these samples, these samples are used in different statistical calculations where it is important to have randomly generated samples (e.g. search engine size estimation or its quality test). However, in search engines like Google, complicated ranking mechanisms violate this randomness assumption and accordingly, generate

uncertainties in calculations. In sampling documents, factors such as chosen query, content of documents and ranking mechanisms affect the probability of a document to be selected and therefore, the random generation of samples. Non-random samples make calculations, based on QBS, doubtful. To deal with these non-random samples, a number of different approaches are introduced in the literature [11, 81, 127]. However, none of these approaches aim at reducing duplicates in samples but keeping calculations free from biases by applying importance and rejection sampling methods to select random samples from non-random ones or by applying formulas on non-random samples to reduce the effects of biases. These approaches are extensively discussed in Chapter 3. 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* that are also known as *topical crawlers*, a crawl starts from a user-provided set of data and collects results only for given topics [109, 128]. 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, the relevance of a page to a given topic is estimated [128]. These crawlers are designed for particular information needs expressed by topical queries or interest profiles [109].

Query expansion Query expansion is the process of reformulating the original query with the goal of getting a better query, a query that is more likely to retrieve relevant documents leading to better top-k recall and precision and therefore, improving retrieval effectiveness [30]. Automatic query expansion techniques can be classified into five main groups based on the used techniques to find the expansion features: linguistic methods, corpus-specific statistical approaches, query-specific statistical approaches, search log analysis and web data [30]. The pseudo-relevance feedback, from category of query-specific statistical approaches, is one of the more interesting approaches for the purpose of this work where it is assumed that terms in retrieved documents are useful for retrieval tasks and expanding the original query. The criteria for selecting among these terms can be based on a number of different features of documents and terms [29, 30, 37, 67, 108].

4.3 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 the available information* 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 Section 4.4, 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 thesis and is suggested as future work. Therefore, this chapter focuses on finding the most efficient approaches to achieve the biggest possible coverage for a given entity in terms of matching documents.

Relevance Looking for an entity, we gather all the pages that contain the terms in that entity by submitting a phrasal query without applying entity detection, extraction and disambiguation. These techniques are considered as future work.

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 number of query submissions. We compare the implemented approaches based on how they 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 4.3.1 describes ideal approaches for which we estimate the number of needed queries in an ideal (simulated) condition. Section 4.3.2 discusses methods in which queries are reformulated by using an external corpus. In Section 4.3.3, the focus is on techniques inspired by pseudo-feedback methods for query expansion. In this method, the extracted content from the previously retrieved documents is the source for a query generation process.

4.3.1 Ideal approaches

The mentioned approaches in this section are desirable or perfect but not realized easily. These are investigated with the sole purpose of improving the comparison of the introduced approaches.

Oracle perfect approach

To achieve a full data coverage on a given entity in a search system with both `#ResultsLimited` and `#RequestsLimited`, a perfect approach returns not only the maximum possible number of documents (l) but also only the unique ones for each request submitting either $Q^{\texttt{Coll}}$ or $Q^{\texttt{Sample}}$. To have a complete coverage in this situation, it is sufficient to send only the $\frac{|R^{\texttt{Coll}}|}{l}$ number of requests. In reality, this is not easily reachable. To do so, you need to know the exact mechanism of a search engine's ranking algorithm. Then, you can divide the collection of that search engine into exactly $\frac{|R^{\texttt{Coll}}|}{l}$ sub-collections. In addition to the knowledge of ranking algorithm, you need additional information. For instance, if a ranking algorithm is based on frequencies, you need to know all the term frequencies beforehand. This kind of information is only accessible when you have a full access to the search engine's index.

Probability based approach

Provided that there is a uniform selection probability for all the documents matching a given query in a search system, we have no bias in selecting documents to return as query results. This is the case only in search engines without a ranking algorithm. Query and ranking biases are direct results of ranking algorithms and the selection criteria of documents to be returned for a given query. 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. Equation 4.2 calculates the estimated number of unique documents and also duplicates among the results of any of the query submissions.

$$|R_n^{\texttt{Cum}}| = |R^{\texttt{Coll}}| - |R^{\texttt{Coll}}| \times (1 - \frac{l}{|R^{\texttt{Coll}}|})^n \tag{4.2}$$

Formula definition Formula 4.2 calculates the number of newly discovered documents , $|R_n^{\texttt{Cum}}|$, from all the submitted queries to a search engine. It is assumed that documents follow a uniform selection probability. In this case, this probability is defined as $\frac{l}{|R^{\texttt{Coll}}|}$. Consequently, the probability of a document not to be selected is defined as $(1 - \frac{l}{|R^{\texttt{Coll}}|})$. Now, the goal is to calculate these probabilities after n number of query submissions (sampling events). A unique document in the n^{th} sampling event means the document is not selected in

$(n-1)$ previously generated samples. With $(1-\frac{l}{|R^{\texttt{Coll}}|})$ as the probability of a document not to be selected in one sampling event, we define $(1-\frac{l}{|R^{\texttt{Coll}}|})^n$ as the probability of a document not being selected in n sampling events. Multiplying this probability in the total number of documents matching a given query ($R^{\texttt{Coll}}$) determines the number of not retrieved documents by the n number of previous query submissions. Subtracting this number from $|R^{\texttt{Coll}}|$ gives the number of all the previously retrieved documents.

Simulation approach

In this approach, we simulate a random selection of documents. In this simulation, random samples are generated to measure the number of duplicates and unique documents in random sampling events.

4.3.2 List-based query generation approach

In these approaches, the terms to be added to a seed query are selected from a list of words. This list is generated from an external corpus and includes the frequencies of terms in that corpus. We extract this list from the *ClueWeb09* data set, which is a web crawl containing nearly 500 million English pages [120]. Selecting terms from a list of terms with their corresponding document frequencies can be performed in different methods. In following, these methods are further explained and studied.

List-based most/least frequent approach

Although primitive, choosing the most or least frequent words from a list are possible options in selecting terms. As the ClueWeb data set is not a topic-specific corpus, the most frequent words from this corpus are highly probable to be also general in all other non topic-specific corpora. However, this is not the case in our application. The targeted collection is based on a given query. In addition, least frequent terms from a web crawl are potentially terms with spelling errors. Therefore, these two options are not considered in our experiments.

Pre-determined frequency based approach

While submitting the most frequent terms increases the chance of reaching a maximum number of returned results and the least frequent ones increases

the probabilities of generating fewer duplicates, it is of a great interest to investigate the likelihood of finding a term frequency which creates a trade-off between these two. To do so, statistical formulas are applicable. If events A and B are independent, the probability of both of them occurring is the product of their occurring probabilities ($P(A\&B) = P(A) \times P(B)$). If we assume query submissions as independent events then, the probability of having an overlap between two queries equals the multiplication of the probabilities of queries. This is shown in Equation 4.3.

$$\frac{|R^{\texttt{Coll}} \cap R^{\texttt{Sample}}|}{s^{\texttt{SE}}} = \frac{|R^{\texttt{Coll}}|}{s^{\texttt{SE}}} \times \frac{|R^{\texttt{Sample}}|}{s^{\texttt{SE}}} \tag{4.3}$$

We solve Equation 4.3 for $|R^{\texttt{Sample}}|$ where $|R^{\texttt{Coll}} \cap R^{\texttt{Sample}}| = l$. This results in Equation 4.4

$$|R^{\texttt{Sample}}| = \frac{l \times s^{\texttt{SE}}}{|R^{\texttt{Coll}}|} \tag{4.4}$$

Through Equation 4.4, with the knowledge of a targeted search engine's index size ($s^{\texttt{SE}}$) and also the number of documents matching a *seed query* (the first query), one can determine the frequency of another query ($|R^{\texttt{Sample}}|$) for which the overlap of this query and the seed query equals l. This means with information about the seed query, returned results and search engine size, a term can be found to formulate a new query returning at least l results. This enables avoiding the permanent presence of the same highly ranked documents among results and creates a higher chance in collecting more new documents in each trial. If the size of a search engine is unknown, as discussed in Chapter 3, its size can be estimated by using generated samples from that search engine.

As pointed out, applying this formula to our case requires information on document frequencies of terms. To access this information from the targeted search system, we should download all its content and count all the document frequencies of terms. If this was possible, there was no need for introducing new approaches. Instead, we can use pre-computed document frequencies of terms from an external corpus. In this work, as we test our approaches on Google, we use ClueWeb09 data set. ClueWeb09 and Google have different sizes. To apply the formula, we include both sizes in the calculations.

For example, in equations 4.3 and 4.4, if we set $s^{\texttt{SE}} = 10^9$, the number of English documents in ClueWeb09 as 5×10^8, $l = 100$ and $|R^{\texttt{Coll}}|$ for a given query as 4×10^5, the following calculation provides us with document frequency of a term that has higher chance to result in samples of our desired

size: $\frac{100}{10^9} = \frac{4\times10^5}{10^9} \times \frac{x}{5\times10^8} \Longrightarrow x = 125000$. Consequently, we select terms with 125000 document frequency from ClueWeb09 data set for the query expansion. This method is called the *LB-FixedFreq.* approach.

4.3.3 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.

FB-most frequent terms based approach

Among the extracted terms from the previously 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 Section 4.3.2. This approach commences by submitting a query to a search engine. Then, having extracted the content of results for that query, the most frequent word in the content is selected to be used in query reformulation. The final steps are adding this most frequent word to the original query and submitting the constructed query to a search engine. With new results obtained from this query submission, these steps are repeated and new queries are formed and submitted. This process repeats until we reach a full data coverage for the seed query. For example, with the goal of reaching full data coverage for the term "Vitol" that is an oil company, the term "Oil" appears as the most frequent term in the returned documents by the search engine. The next step is submitting "Vitol"+"Oil". This technique is called *FB-MostFreq.* approach.

FB-least frequent terms based approach

In this method, contrary to FB-MostFreq., the least frequent terms are selected. The reasoning behind this difference is the struggle to minimize the number of duplicates among the generated samples. We expect that the least frequent terms have high chance of obtaining fewer duplicates. For example, in the first step to cover all related documents to the term "Vitol", the term "damit", which is a German word, is selected to be used in reformulating the query. This technique is referred as *FB-LeastFreq.* approach.

FB-least from last approach

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

FB-fixed frequency based approach

In Sections 4.3.3 and 4.3.3, 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 fewer duplicates potentially. To introduce a balanced approach that includes both these cases, Equation 4.3 is applicable. This equation calculates a specific frequency that has higher chance to result in samples of our desired size l. However, we can use Equation 4.3 to select terms only from an external corpus with a pre-calculated list of terms and frequencies. To apply this formula to a feedback-based method, it is required to determine the document frequency of a term in retrieved documents ($\texttt{T.D.F.}^{\texttt{FeedbackText}}$) that corresponds to the resulted frequency $|R^{\texttt{Sample}}|$ from Equation 4.3. To do so, we can apply Equation 4.5.

$$\texttt{T.D.F.}^{\texttt{FeedbackText}} = \frac{|R^{\texttt{Sample}}| \times |R^{\texttt{Cum}}|}{s^{\texttt{SE}}} \tag{4.5}$$

In this equation, $|R^{\texttt{Sample}}|$ is resulted from Equation 4.5. With the number of retrieved documents as $|R^{\texttt{Cum}}|$ and the size of a search engine as $s^{\texttt{SE}}$, $\texttt{T.D.F.}^{\texttt{FeedbackText}}$, which stands for a term's document frequency in feedback text, is determined for the next term selection. As this approach investigates a trade-off between the extreme cases of submitting the most and least frequent terms, we expect the terms with this frequency to have higher

a chance to result in samples of a desired size and with few duplicates. This method is referred as *FB-FixedFreq*.

4.3.4 Combined list-feedback based approach

In the list-based methods, the terms are selected from an external corpus considering their frequencies. 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, *Comb.LB-FB* approach selects terms that are present in the list of terms from an external corpus and also in the previously retrieved documents. For example, as the next query after submitting "Vitol", we choose a term appearing in both the retrieved content and list from ClueWeb09. This helps to have both relevance and frequency in one method. This technique is called *Comb.LB-FB*.

4.4 Experiments and results

4.4.1 Experiments settings

Test search engine In these experiments, Google, as the biggest web search engine with one of the most complicated ranking algorithms, is considered as our test search engine. Targeting Google does not limit our findings since the only necessary feature of a website for applying any of the suggested approaches is the support of a keyword-search interface. We believe if the suggested approaches work for Google, they can also work for a wide range of websites. Although Google is used only as an example of a search engine in our work, 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 [60]. Through our experiments, we noticed that not more than 500 results are accessible for any submitted query. This number was dynamic but it was less than 500 in all our experiments. 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 websites. 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", "Fireworks Disaster", "Ed Brinksma" and "PhD Comics"). 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 sizes of matching documents ranging from 2×10^4 to 5×10^5.

Relatedness judgement To assess an approach in achieving a full data coverage for 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 returned documents by a search engine for an entity (R^{Cum}) as retrieved data. We judge all the retrieved documents that contain the searched keyword as a relevant document. There are other possible options that are mentioned as future work. As all the documents contain the keywords, retrieving more documents increases the chance of achieving more completeness.

Evaluation metric Assessing different approaches for one entity is straightforward by comparing the R^{Cum}s of all tested approaches. However, the general performances of these approaches on all the entities can not be evaluated through only the comparison of their R^{Cum}s. With different numbers of results for entities, comparing $R^{\text{Cum}}_{n_{\text{Entity1}}}$ and $R^{\text{Cum}}_{n_{\text{Entity2}}}$ of the same method does not reveal much information about its overall performance. While a small number of queries can cover all documents for a small results set, for a bigger collection, that number of queries can just retrieve a very small part. Therefore, we evaluate each method for a given entity by its distance from the performance of the probability-based approach (Section 4.3.1) for that entity. This is calculated through Equation 4.6. These distances are calculated for all the other entities in the entities test set and averaged to represent the general performance of that approach.

$$\texttt{Performance} = \frac{|R^{\text{Cum}}_{n_{\texttt{Prob.Appr}}}| - |R^{\text{Cum}}_{n_{\texttt{Test.Appr}}}|}{|R^{\text{Cum}}_{n_{\texttt{Prob.Appr}}}|} \times 100 \tag{4.6}$$

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 of this chapter, 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.

4.4.2 Results

In this section, the results of applying the introduced approaches in Section 4.3 for the test entities (Section 4.4.1) are presented. To establish a comparison baseline, in Figure 4.2, Oracle Perfect, Probability-based and simulation approaches (4.3.1) 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 a maximum size and no duplicates. The Probability-based method performs worse than Oracle Perfect but the same as the simulation.

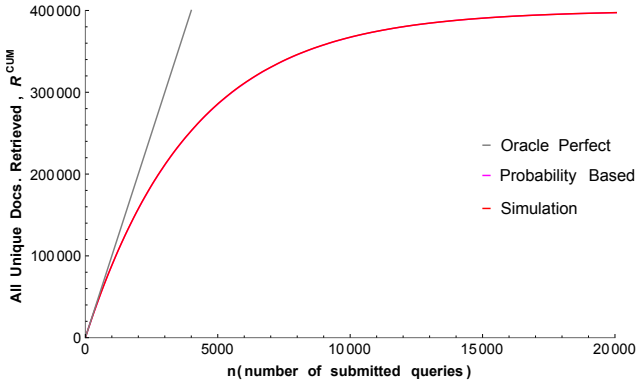


Figure 4.2: Retrieved documents (R^{Cum}) for ideal approaches. The lines for simulation and probability-based approaches overlap completely.

The results of running the approaches of Section 4.3.3 are shown in Figure 4.3. Among the FB-Based methods, Comb.LB-FB has more samples of the maximum size and fewer duplicates than other techniques. This is shown in Figures 4.4 and 4.5. In Figure 4.4, the top right corner of each graph is the most desirable place. The points in this corner represent samples of a maximum size and with fewer duplicates. Figure 4.5 compares the sizes of samples and duplicates among them for all the FB-based approaches. While the top part of Figure 4.5a, the *Sample Sizes* figure, is more desirable, the points in the top part of Figure 4.5b, the *duplicates* graph, are less desirable. Figure 4.3 compares ap-

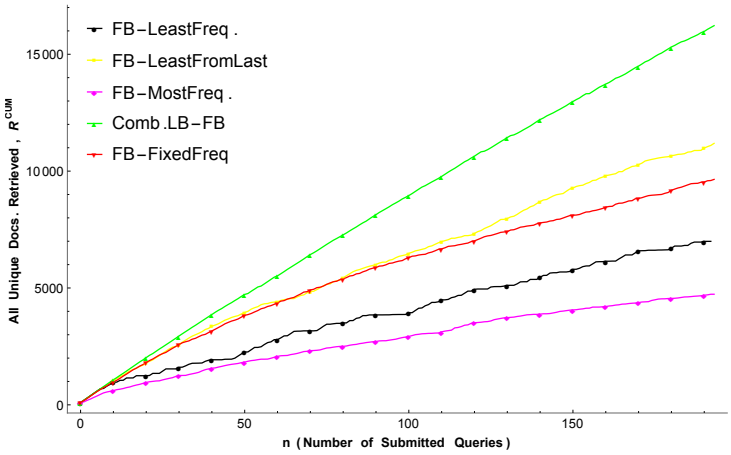


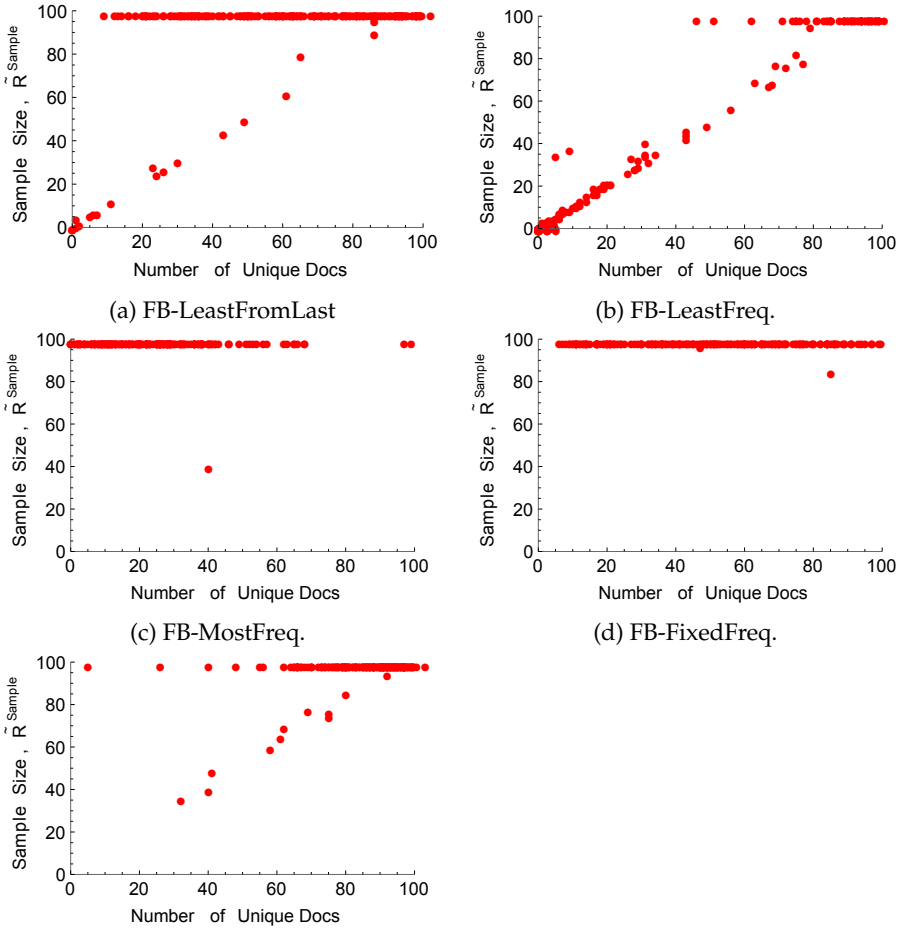
Figure 4.3: Retrieved documents (R^{Cum}) for FB-based approaches for one entity (Vitol)

proaches for one entity, whereas Figure 4.6 compares the average performances of all the approaches for all the entities in the test set. These average performances are calculated through Formula 4.6 in Section 4.4.1. As it is observable from this figure, there are big gaps between each approach and the estimated probability. This emphasizes the effect of ranking algorithms of search engines.

4.4.3 Analysis

As illustrated in Figures 4.4 and 4.5, the key to success (having more data coverage) is bigger samples with fewer duplicates. However, there is a trade-off between these two goals. Bigger samples increase the chance of more duplicates. Instead, to reach fewer duplicates, smaller samples are helpful. In FB-MostFreq. technique (Section 4.3.3), samples are big but include a lot of duplicates. In FB-LeastFreq. (Section 4.3.3), this is exactly the other way around.

One way to tackle the challenge of achieving this trade-off is to find a specific term frequency to select the next terms to form queries. In LB-FixedFreq. technique (Section 4.3.4), different frequencies are applied. In our experiments, we observed that the low and high frequencies yield to a worse coverage than submitting words with a frequency derived from Formula 4.3. Another way to achieve this trade-off is to counteract the biases of search systems in select-



(a) FB-LeastFromLast

(b) FB-LeastFreq.

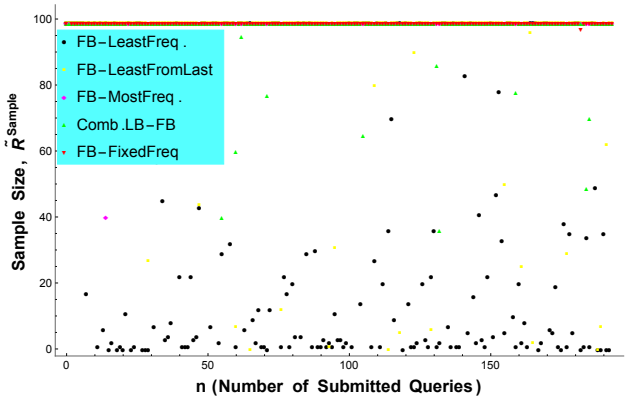
(c) FB-MostFreq.

(d) FB-FixedFreq.

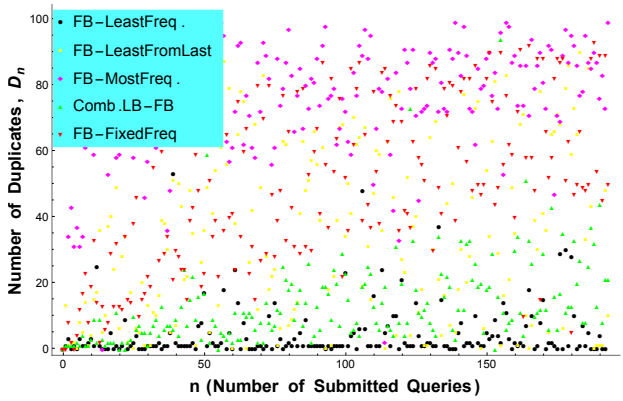
(e) Comb-LB-FB

Figure 4.4: Sample sizes vs. unique documents in generated samples by the FB-based methods

ing documents like PageRank-generated bias in Google. The introduced FB-ListFromLast technique, which is based on this bias removal idea, produced better results than FB-MostFreq and FB-LeastFreq. However, this approach could not outperform Comb.LB-FB. In Figures 4.4 and 4.5, it is observable that



(a) Sizes of generated samples



(b) Duplicates in generated samples

Figure 4.5: Sizes and duplicates of generated samples for FB-based methods

the number of duplicates and sizes of samples for Comb.LB-FB. are more desirable than the ones in FB-ListFromLast.

Among all the introduced approaches, Comb.LB-FB performs the best. This method intends to reduce the number of small samples with selecting terms with a specific frequency, extracted from an external corpus and refined by checking their presence in the previously downloaded documents. This causes 10 percent better documents coverage, in average for all the submitted entities.

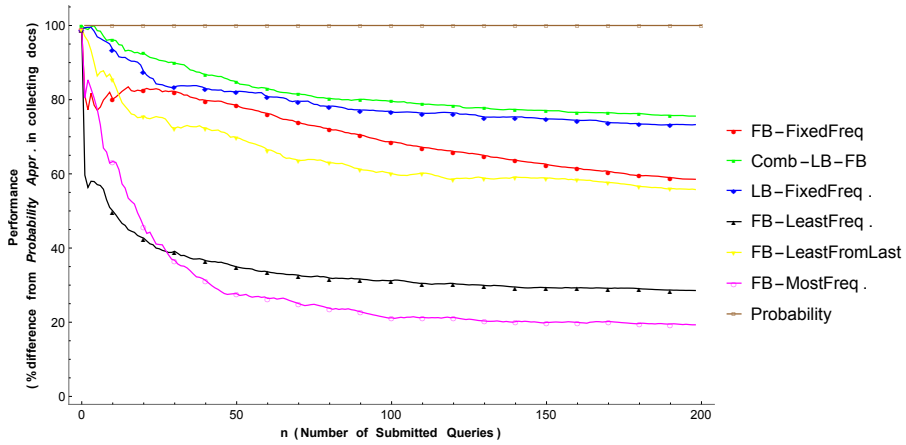


Figure 4.6: Average performances of approaches for all the test set entities

4.5 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 that result in the maximum possible number of results with the minimum possible number of previously downloaded documents (duplicates) among them. To have this success factor, we suggested different approaches based on different frequencies and possible dependencies. From these approaches, the Comb.LB-FB method, which analyzed the retrieved results and combined it with information from a large external corpus, performed the best.

Future work In addition to considering frequencies of words and their presence in retrieved documents to select the best next query to submit, there are a number of issues such as terms distribution in returned documents, their distances and dependency of words to all the previously submitted queries that can help to counteract the generated biases by search engines. In future work, we can focus on experimenting the effects of these factors. As another next step, the retrieved documents can be disambiguated, analyzed and entities in those documents can be extracted. For example, searching Vitol on a search engine can result in different topics from a company to a person. Analyzing

the returned results by different techniques like documents clustering helps to refine the retrieved documents and analyze only the ones that are of our interest.

CHAPTER 5

EFFICIENT DEEP WEB CONTENT MONITORING

Given the fast evolving nature of the web, in focused web harvesting, how can one efficiently harvest the changed content?

This chapter is based on [86].

Web content changes rapidly [95, 97]. In *Focused Web Harvesting* [84] which aim it is to achieve a complete harvest for a given topic, this dynamic nature of the web creates problems for users who need to access a set of all the relevant web data to their topics of interest. Whether you are a fan following your favorite idol or a journalist investigating a topic, you may need not only to access all the relevant information but also the recent changes and updates. General search engines like Google apply several techniques to enhance the freshness of their crawled data. However, in *focused web harvesting*, we lack an efficient approach that detects changes for a given topic over time. In this chapter, we focus on techniques that can keep the relevant content to a given query up-to-date. To do so, we test four different approaches to efficiently harvest all the changed documents matching a given entity by querying a web search engine. We define a document with changed content or a newly created or removed document as a changed document. Among the proposed change detection approaches, the *FedWeb* method outperforms the other introduced approaches in finding the changed content on the web for a given query with 20 percent, on average, better performance.

5.1 INTRODUCTION

Web content is constantly and rapidly changing [21, 31, 35, 95, 97, 137]. This dynamic nature of web data creates difficulties for users who need to access a complete collection of documents relevant to their topics of interest. For example, as a fan following your favorite idol or a business analyst studying a stock market, you want to access not only *all* but also the *latest* information relevant to your topics of interest. You need to keep your collection up-to-date over time.

To monitor data, first, we need a complete coverage for a given topic. To reach this complete coverage, accessing a crawl of the whole web is an optimum solution. However, accessing such a crawl seems impracticable even for big companies and governments. As an alternative, *focused web harvesting* techniques are applicable. We defined focused web harvesting as harvesting all documents matching a given entity by querying a web search engine in Chapter 4 [84]. For instance, information about "Bernie Sanders", "Islamic State" or "Golden Ball Award" are retrieved from indexed data in general search engines or hidden data behind web forms by submitting a stream of queries and retrieving their returned results. Queries are formed by adding terms to a seed query with the goal of returning more unique documents with respect to the imposed limitations by search engines on the number of submitted queries by a user and the number of returned results he can view [84].

These imposed limitations by search engines also affect having an up-to-date data collection. Through applying focused web harvesting techniques, users can collect relevant information to their topics of interest and monitor its changes over time by re-issuing queries. However, considering the imposed limitations by search engines, we need efficient approaches to detect the changed content on the web without recollecting all the previously harvested documents. We need approaches that facilitate efficient re-harvesting processes. To attain these approaches, we study the applied techniques in crawlers and big search engines to increase the freshness of their indices. We also explore their algorithms for detection of changed, duplicate and near-duplicate pages. We investigate deep web harvesting techniques, especially focused web harvesting, to find methods applicable in detecting changed content on the web for a given topic. Based on these studies, we propose approaches that are based on adapting a query generation mechanism to return more desirable documents. This work proposes solutions for efficient retrieval of changed content on the web in the domain of focused web harvesting.

Contributions As the first contribution of this work, we study the change rate of *FedWeb data sets* from two different years [44]. In this study, we analyze change rates of 150 websites from 24 categories. As the second contribution, we study four different methods to find the most efficient approach to retrieve changed documents on the web that are relevant to a given topic. We test these approaches on our test search engine and report the results. The results show that we can improve the retrieved changed content by at least 20 percent by submitting the same number of queries but the ones that are more likely to occur in changed documents.

Outlook In Section 5.2, we study the literature for applied methods by general search engines to keep their indices up-to-date and increase their *index freshness*. We also study the literature for methods to detect changed, duplicate or near-duplicate HTML pages in Section 5.2.4. Section 5.3 discusses the change rate of websites and their categories. In Section 5.4, the proposed techniques for efficient re-harvesting are introduced. The results of testing these methods on our test set is presented in Section 5.5 and analyzed in Section 5.6. Section 5.7 draws conclusions and discusses further future work.

5.2 Literature study

In this section, first, we study applied techniques in web crawlers to update their local views of the web to reflect changes occurring on the web. From web crawlers and search engines domain, we explore the applied techniques to discover new pages, keep their indices up-to-date and detect similar pages. We also study the web harvesting literature to investigate methods for obtaining and monitoring deep web data.

5.2.1 Web crawling and focused web harvesting

Search engines use web crawlers to collect pages from the web [31, 61]. In general, a web crawler starts with a number of initial URLs known as seed URLs. The corresponding web pages to these URLs are fetched (if *robots.txt* file allows) and parsed to extract hyper-links in their content. The web pages of these hyper-links go through the same process of parsing and extracting. These steps are repeated until enough URLs to meet crawler's goals are visited [31, 61, 114].

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 page deletions have already changed the web content [10]. According to an estimation [114, 138], the surface web currently contains more than 4.7 billion web pages with an enormous growth in comparison to the estimated 800 million pages in 1999 by Lawrence *et al.* [95]. In addition to removal or creation of pages, the change in content of web pages has a very high rate. The frequency of a web document's change has been studied in previous works reporting the change rate of web pages between a day to a year, varying dramatically from site to site and object to object [21, 31, 137]. The most popular objects have higher rates of change with changes that can be modest or significant [31, 137]. Cho *et al.* [35] studied how often a page changes over different domains (e.g. *.net*, *.org*, *.com*, *.edu* and *.gov*) [35]. Wolf *et al.* [137] showed that 23% of web pages and 40% of commercial web pages change daily, demonstrating how quickly a search engine index gets out-of-date [137].

Dealing with this ever-changing content is a challenging task for search engines and needs extra network resources [31, 35]. The following section presents different applied strategies to keep this extra load on network as minimized as possible.

5.2.2 Crawling strategies

Different strategies are used for web crawling [106, 114]. In this chapter, we divide crawlers into two main classes, general crawlers and specific crawlers. We define general crawlers, also known as unfocused crawlers, as crawlers targeting all pages on the web, while specific crawlers limit the targeting of URLs based on focusing on a specific domain (domain specific or focused crawlers), a specific topic (topic specific crawlers), 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 [106, 114].

To detect changes in web pages, some crawlers apply simple re-visiting policies like uniform policy (re-visit all pages regardless of their change rates) or proportional policy (pages that change more are re-visited more) [31]. Although these processes are simple, they add extra network load (too often changing pages in proportional policy) and consider all pages on the web to be 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 a web

page, its change frequency, its intrinsic quality, its popularity in terms of links or visits and even its URL. Some research work focus on modeling the change frequency of a web page by a *poisson* process [35], as a *renewal* process [21] or as *stochastic marked points* processes [137].

5.2.3 Deep web harvesting

Recent studies on accessing deep web data mainly focus on harvesting websites requiring form submissions [6, 25, 71, 105], studying the web forms to achieve an efficient data access approach. For example, Madhavan *et al.* [105] investigate how to identify the values and types of different web 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 domain knowledge.

A deep web data access is mainly performed through *query based sampling* referred to as *QBS*. In QBS, by sending a query to a search engine, the set of returned results is considered as a sample of documents [19]. To form these queries, *query expansion* techniques are applied. Query expansion is the process of reformulating a given query with the goal of getting a better query, a query that is more likely to retrieve relevant documents [30]. One of the query expansion techniques is pseudo-relevance feedback that assumes that terms in retrieved documents are useful for expanding the original query. The criteria for selecting these terms can be based on a number of different features of documents and terms [30].

The idea of *monitoring* deep web content is discussed in several papers [1, 17, 20, 78, 98] which mostly focus on re-running web data access approaches. Boncella *et al.* [20] provide an overview of how monitoring web data can be useful for competitive intelligence such as detecting cognitive hacking attacks on companies. Abiteboul [1] also discusses different aspects of monitoring both surface web and deep web data but without suggesting practical approaches. None of these studies focus on *efficiency* in monitoring change in deep web sources while highly limited resources of harvesting processes impose the need for efficient techniques.

Recent business applications for monitoring web data such as *BrightPlanet*

Deep Web Monitor[1] (example of a deep web harvester) and *Google Aerts*[2] (example of a surface web monitoring tool) perform similarly based on a regular re-running of their indexing or harvesting processes. *Google Alerts* alerts users for their topics of interest when new results for a given query are added to Google. Bharanipriya *et al.* [18] also discusses a number of harvesting tools that can monitor web pages by following the same principle of regular revisiting.

5.2.4 Changed, near-duplicate and duplicate pages

In comparing pages to detect changed content on the web, we can label pages either as duplicate, near-duplicate or changed. Two documents with exactly the same content are referred as duplicate pages. However, in case of small dissimilarities between their content, they are identical to a remarkable extent and known as *near-duplicate* pages. For example, two pages with a few different words, different formatting but similar text or some typographical errors, plagiarized documents, different versions of a document and diverse file types can be considered as near-duplicate pages [111].

In collections of web documents, different techniques are applied to find similar pages. Based on the introduced classifications by Manku *et al.* and Mudhasir *et al.* [107, 111], we propose a more comprehensive classification of near-duplicate detection techniques in Figure 5.1. As shown in Figure 5.1, near-duplicate detection techniques have different feature sets like URL, text syntactic, text semantics, structure and connectivity. In URL-based techniques, instead of page content, only URLs are used to determine if a page is a near-duplicate. In connectivity-based approaches, linkage structure of the web is probed to find similar pages. The basic idea is that similar pages have similar incoming links [107]. In structure-based techniques, the structural similarities of web pages are studied to identify schemas and templates of pages. In text-semantic based approaches, page semantics are considered. The text-syntactic-based approaches focus on analyzing the syntax of a page content. In this analysis, a document is represented by different feature sets. 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"[3] [22], with $k = 4$, are "a rose is a", "rose is a rose" and "is a rose is".

[1] https://www.brightplanet.com/solutions/deep-web-monitor
[2] https://support.google.com/alerts
[3] A sentence written by Gertrude Stein as part of the 1913 poem *Sacred Emily*.

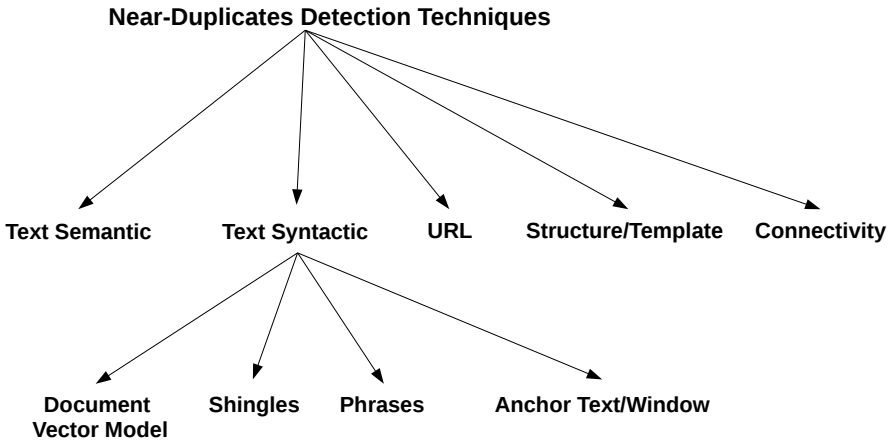


Figure 5.1: Classification of near-duplicate detection techniques

In a document-vector model, by using traditional information retrieval (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 [66]. In Phrases-based techniques, phrases in a page are detected and considered as terms in a document vector [107].

To apply these techniques on big data collections, crawlers need to compress the feature sets for fast comparisons. Mod-p shingles [73], Min-hash for Jacquard similarity of sets [24], Signatures/fingerprints over IR-based document vectors [36,88] and Checksums [122] are among the techniques applicable for compressing a feature set [107].

To measure the 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, that are required to change one text into another one, is calculated. For example, the distance between “sittin” and “sitting” is an insertion of “g” at the end.

Detection of changed content In our experiments, we define two documents as near-duplicates by computing a Jacquard coefficient with a preset thresh-

old of 0.9. We fix k as the shingle size. Let $\texttt{Shingles}(\texttt{Doc}_A)$ be the set of shingles in Doc_A and $\texttt{Shingles}(\texttt{Doc}_B)$ the set of shingles in $\texttt{Doc}_B$. We compute Jacquard coefficient through $\frac{|\texttt{Shingles}(\texttt{Doc}_A) \cap \texttt{Shingles}(\texttt{Doc}_B)|}{|\texttt{Shingles}(\texttt{Doc}_A) \cup \texttt{Shingles}(\texttt{Doc}_B)|}$. We extract the texts from web pages with similar URLs, calculate shingle sets of each document and save the fingerprints of shingles. For each pair of documents, if the Jacquard coefficient does not exceed the predefined threshold, we consider the pages as changed documents.

5.3 RATE OF WEB CONTENT CHANGE

Literature studies for the web content change [21, 31, 35, 137] mainly focus on investigating a collection of web pages. Although there are studies investigating change rates in domains (*.net*, *.org*, *.com*, *.edu* and *.gov*), commercial web pages and publications, we intend to compare change rates of a wide range of search engines and their topical categories. We want to study the change rates of different search engines as access points to web data. For our studies, different crawls of the web in different time points are required. In our experiments, we select the FedWeb collection that provides two crawls of 150 search engines from 24 different categories.

Fedweb The FedWeb data set [44] is designed for research in federated web search. The authors provide two official TREC versions called *FedWeb13* and *FedWeb14*. Each version consists of the top-10 search results of sampled queries, as well as a set of test topics, on about 150 search engines. This allows us to study the change rates of a wide range of categories and websites. Therefore, we choose FedWeb for our analysis.

Change definition In our experiments, we define a change on the web as finding a new URL or a previously visited URL with changed content. If there is a change only in rankings of returned results for the same queries, 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.

5.3.1 Change rate of websites in FedWeb

Demeester *et al.* [44] crawled around 150 search engines in two different time points in 2013 and 2014. To analyze the change rate of search engines in FedWeb, we analyzed the results for the same queries in both 2013 and 2014 crawls

and counted the number of new and similar URLs among them. We also investigated the content change for similar URLs applying Shingle Jacquard technique. The results of these comparisons are shown in Figure 5.2. This figure also depicts the number of queries that returned results after their submissions to search engines and therefore, considered in our evaluation.

As our samples are search engine results, if a returned page for a submitted query in 2013 is no longer among the returned results for that query in 2014, there is a chance that page is not changed but not returned due to a different ranking of results. To reduce the number of these pages and their effects in our change rate analysis, when comparing results from both years for a query, we search all the returned documents in 2013 and 2014 instead of considering only the returned results for that query.

5.3.2 Change rate of categories in FedWeb

FedWeb collections divide search engines into 24 categories covering a wide range of domains [44]. These categories and the number of websites from each category are mentioned in detail in Table 5.1. The averaged change rates of categories with their corresponding minimum and maximum values are depicted in Figure 5.3.

Table 5.1: 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
Games	6	Shopping	9
General	10	Social	3
Health	13	Software	3
Jobs	5	Sports	7
Jokes	2	Tech	8
Kids	9	Travel	2
		Video	14

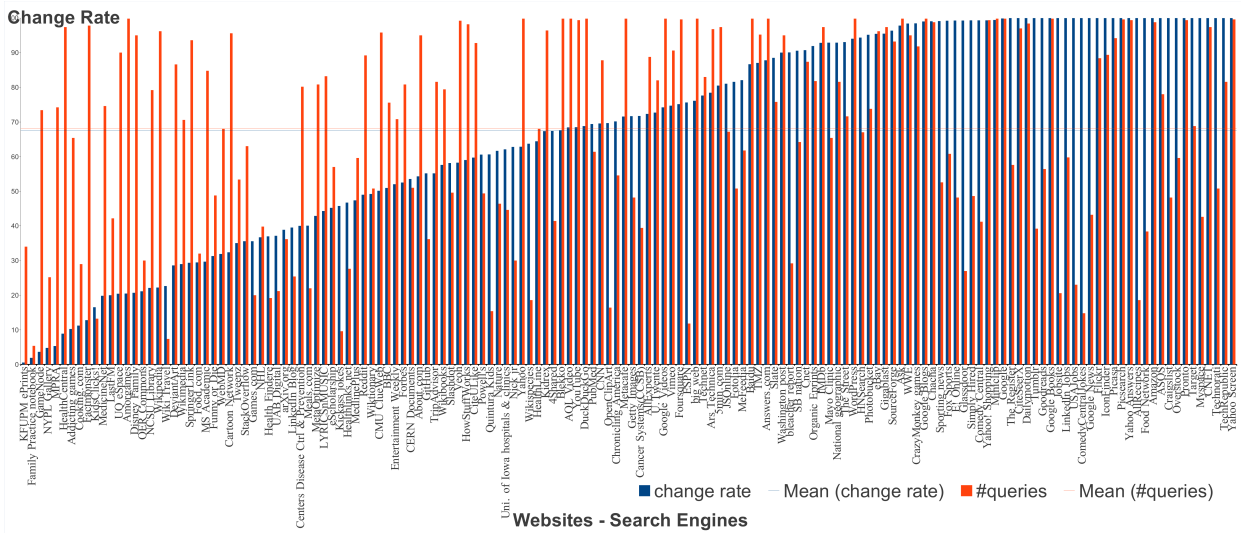


Figure 5.2: Change rate of search engines in FedWeb. For each website, its change rate is calculated considering queries that returned results in both FedWeb13 and FedWeb14. The number of these queries for each category is also presented. The mean change rate of all websites is also shown in this figure.

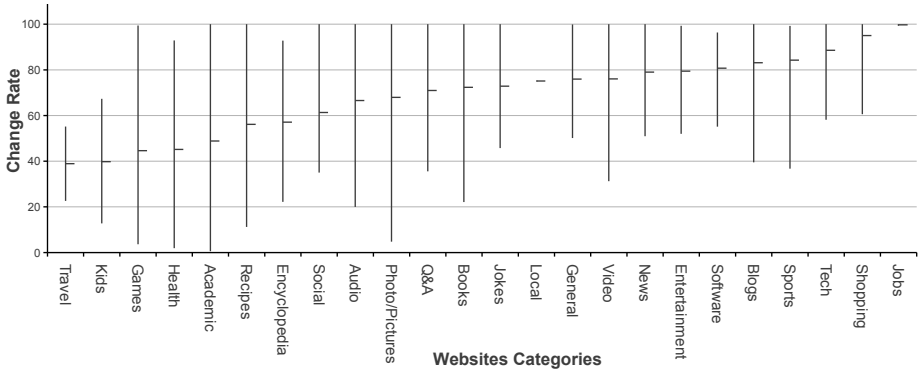


Figure 5.3: Change rates of categories in FedWeb

5.3.3 Change rate analysis

Figures 5.3 and 5.2 show that a change rate differs from website to website and category to category. Although some categories are not thoroughly represented in the FedWeb collection (refer to Table 5.1), we can still draw some conclusions. As shown in Figure 5.3, some categories like *Jobs, Tech, Shopping, Sports and Blogs* show near 100 percent change while websites in *Travel and Kids* categories observe much less changed content but still more than 40 percent.

Among the analyzed search engines in Figure 5.2, the websites in Shopping, Technology, Software, Entertainment and News categories are always among the top changing collections. One of the websites with the lowest change rate is a university electronic prints. The reason behind the observed low change rate of this website can be either not maintaining the website, no new publications for the test queries or the small size of the website which makes it less likely to get new results for the queries.

To conclude, we can claim that content of FedWeb collection changed, on average, around 70 percent as shown in Figure 5.2. However, the change rate differs substantially per domain and website as shown in Figures 5.2 and 5.3.

5.4 Solution for web content monitoring

As the first step to monitor web content changes, in focused web harvesting, we need a complete collection of relevant documents at hand. We use the in-

troduced *Comb.LB-FB* method in Chapter 4 to collect the first set of relevant documents. Comb.LB-FB is an efficient approach for focused web harvesting based on using a predetermined list of words and extracted content from previous query search results.

After a predetermined time, set to two weeks in our experiments, we perform the second harvest. We re-harvest documents with the same queries to have a baseline for detecting changes. This second re-harvest is called *SecondCrawl* in our experiments. As mentioned in Section 5.3, we define a changed document as a document with changed content compared to its previously retrieved version or a completely new retrieved document. Based on this definition, we detect and count the changes documents in SecondCrawl. We compare the content of two pages based on the discussed Shingle-Jacquard technique in Section 5.2.4.

In this work, our goal is to implement approaches that detect and harvest changed pages efficiently. Our proposed approaches are based on expanding a seed query (given entity). To expand a seed query, we favor terms that, in general, generate fewer duplicates and bigger samples and result in more changed and fewer unchanged pages. To find these terms, we focus on approaches inspired by pseudo-feedback methods for query expansion. In this method, the extracted content from previously retrieved documents is the source for a query generation process. Therefore, we employ different strategies such as analyzing the extracted content of returned results by a search engine and also lists of words from external corpora to expand a seed query. The proposed approaches are discussed in the following sections.

5.4.1 Most frequent from new documents

In this method, the terms to form queries are selected from the content of previously retrieved pages. From these retrieved pages, the ones that are detected as changed pages are used for terms selection. Among the extracted terms from these changed documents, we select the terms with higher frequencies as they increase the chance of returning more results and therefore, bigger samples.

This method submits a seed query to a test search engine and detects the changed documents. The most frequent word in those changed documents is selected to be added to a seed query. The final step is submitting this formed query to a search engine. With the 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 returned changed documents. The next

step is submitting "PhD Comics"+"graduate". This approach is called *MostFreq* method.

5.4.2 Least frequent from new documents

The *LeastFreq* approach performs on the same basis of MostFreq but instead of selecting the most frequent terms, the least frequent ones are selected to reformulate queries. These least frequent terms have higher chances of reducing duplicates among changed documents.

5.4.3 Combined list and feedback from new documents

The most and the least frequent terms represent extreme cases of increasing chances of having either bigger samples or fewer duplicates. To reach a balanced approach, we need to determine a specific frequency between these two extreme cases for a term selection.

To do so, through Equation 5.1, with the size of a search engine $\texttt{size}^{\texttt{SE}}$ and the number of matching documents to a seed query $|R^{\texttt{Coll}}|$, we can calculate a frequency $|\texttt{Sample}^{Q}|$ to select terms from an external corpus. If the search engine size is unknown, we can estimate its size through approaches introduced in Chapter 3 [81]. For further information about this equation, you can refer to Equation 4.3.

$$\frac{|R^{\texttt{Coll}} \cap \texttt{Sample}^{Q}|}{\texttt{size}^{\texttt{SE}}} = \frac{|R^{\texttt{Coll}}|}{\texttt{size}^{\texttt{SE}}} \times \frac{|\texttt{Sample}^{Q}|}{\texttt{size}^{\texttt{SE}}},$$
$$\texttt{where} \quad |R^{\texttt{Coll}} \cap \texttt{Sample}^{\texttt{Query}}| = l \quad \longrightarrow \quad |\texttt{Sample}^{Q}| = \frac{l \times \texttt{size}^{\texttt{SE}}}{|R^{\texttt{Coll}}|} \tag{5.1}$$

For example, with $\texttt{size}^{\texttt{SE}} = 10^9$ as the size of a targeted search engine, 5×10^8 as the number of English documents in an external corpus (e.g. ClueWeb09), $l = 100$ as the number of returned results that can be accessed and $|R^{\texttt{Coll}}| = 4 \times 10^5$ as the number of matching documents to a given seed query, through $\frac{100}{10^9} = \frac{4\times10^5}{10^9} \times \frac{\texttt{freq}}{5\times10^8}$, $\Longrightarrow \texttt{freq} = 125000$ in which $\texttt{freq}$ represents the frequency used for selecting terms from an external corpus [84].

The suggested approach 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 expand a given seed query [84].

To apply this approach to retrieve changed documents, instead of selecting terms from the previously retrieved content, a term is used for query reformulation if it appears in the changed content and it is also present in the extracted list from an external corpus. This technique is called *Combined* approach.

5.4.4 FedWeb

In *FedWeb* method, we analyze different versions of crawls of the web collected in different time points. The goal of this analysis is to detect changed documents and find the list of most representative words of this set of changed documents. We analyze two different versions of FedWeb collection to find a list of words that are more common in changed documents than unchanged ones. To expand a seed query in FedWeb approach, we choose a term from a list of words that are representatives of the changed documents in FedWeb collection.

To find a list of the most representative terms for changed documents, we apply a *Transformed Weight-normalized Complement Naive-Bayes (TWCNB)* classifier [124], implemented by Mahout [54], that seeks to maximize term weights on the likelihood that they belong to one class and do not belong to other classes [124]. In this classifier, documents are represented as vectors. The set of these vectors is shown as $\vec{d} = (\vec{d_1}, ..., \vec{d_n})$. Each document vector $\vec{d_1}$ has a label y_i. For any document in this set, d_{ij} is the count of word i in document d_j. By defining a smoothing parameter α for all words in the vocabulary and applying TF-IDF transformation and L2 length normalization, we can assign each term with a corresponding weight as shown in the following steps [124].

As the fist step, TF and IDF transformations and L2 length normalization of $\vec{d}$ are calculated by applying formulas in Equation 5.2. In Equation 5.2, i and m are counters for terms and j and k represent document numbers. In this equation, we define δ_{ij} as 1 if word i occurs in document j and 0 otherwise.

$$\texttt{TF}(d_{ij}) = \log(d_{ij} + 1) \qquad (\texttt{TF transformation}) \quad (5.2a)$$

$$\texttt{IDF}(d_{ij}) = \texttt{TF}(d_{ij})(\log \frac{n}{\sum_{k=1}^{n} \delta_{ik}}) \qquad (\texttt{IDF transformation}) \quad (5.2b)$$

$$\texttt{LN}(d_{ij}) = \frac{\texttt{IDF}(d_{ij})}{\sqrt{\sum_{m=1}^{t} (\texttt{IDF}(d_{mj}))^2}} \qquad (\texttt{Length normalization}) \quad (5.2c)$$

As the second step, to train a classifier for $(\vec{d}, \vec{y})$, $\vec{y} = (y_1, ..., y_n)$, we calculate

the weight w_{ci} of term i for label c, as shown in Equation 5.3. In this equation, α is the smoothing parameter and is set as $\sum_{i=1}^{t} \alpha_i$ with $\alpha_i = 1$ for all words. $\hat{\theta}_{ci}$ is the estimated probability that term i occurs in class c and do not belong to other classes.

$$\hat{\theta}_{ci} = \frac{\sum_{j:y_j \neq c}^{n} d_{ij} + \alpha_i}{\sum_{j:y_j \neq c}^{n} \sum_{m=1}^{t} d_{mj} + \alpha} \tag{5.3a}$$

$$w_{ci} = \log \hat{\theta}_{ci} \tag{5.3b}$$

$$w_{ci} = \frac{w_{ci}}{\sum_i |w_{ci}|} \qquad \texttt{(Weight normalization)} \tag{5.3c}$$

To train this classifier, we need documents with assigned labels. We define two changed and unchanged labels. We compare the content of documents through Shingle-Jacquard text comparison technique, described in Section 5.2.4. Based on the results of these comparisons, documents are assigned with changed or unchanged labels. In our experiments, this step resulted in 45352 documents in changed and 45217 in unchanged classes. Using this set of documents with assigned labels, we can perform the training task. Our trained classifier had an accuracy of $\%86.91$. When the training is done, the classifier calculates the list of representing terms for each category [124]. In these calculations, we assign the smoothing parameter α as 1.

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 *FedWeb* approach.

5.5 Experiments and results

In our experiments, we run the proposed approaches on a test search engine to detect changed documents based on our change definition.

5.5.1 Experiments settings

We test our approaches on a real search engine. In these experiments, Google 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

many entities. Although we target 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, around $30,000$ queries were submitted to download information for four different entities ("Vitol", "Ed Brinksma", "PhD Comics", and "Fireworks Disaster") for the first round. These entities represent diverse types; Company, Person, Topic and Event. In addition to difference in type, we cover entities with the estimated numbers of matching results ranging from 2×10^4 to 5×10^5. These numbers are based on Google search estimates of matching documents.

Evaluation metric To assess approaches for returning changed documents, we need a metric that considers the differences in numbers of matching documents and change rates of entities. To do so, in Equation 5.4, we average the percentage of changed documents that a given approach *Appr.* returns for all the entities through dividing the number of returned changed documents by that approach for an entity E, $\texttt{CDocs}_{\texttt{Appr.}_{E_i}}$, by the total number of found changed documents for that entity $\texttt{TotalCDocs}_{E_i}$. This division is repeated for all the entities in the test set and averaged.

$$\texttt{Evaluate(Appr.)} = 100 \times \frac{\sum_{i=1}^{|\texttt{TestSet}|} \frac{|\texttt{CDocs}_{\texttt{Appr.}_{E_i}}|}{|\texttt{TotalCDocs}_{E_i}|}}{|\texttt{TestSet}|} \tag{5.4}$$

5.5.2 Results

In this section, the results of applying the introduced approaches in Section 5.4 to the test entities (Section 5.5.1) are presented. To establish a comparison baseline, we send the exact same queries from the previous crawl after two weeks and refer to it as *SecondCrawl*. The idea is to see the amount of occurred changes on the web for the exact same information needs. In Figure 5.4, the performances of all the introduced approaches in Section 5.4 in retrieving changed documents are compared. In this figure, for each submitted query, the defined evaluation metric in Equation 5.4 is calculated for all the approaches. This equation averages performances of an approach over all the entities for each query submission. For each query, the number of retrieved changed documents from all the previously submitted queries, up to and including that query, is divided by the total number of changed documents accumulated for a given seed query (test set entity) through all harvesting tasks. This is calculated for all the entities and averaged. This average is defined as the performance of an approach in retrieving changed documents for all the tested entities. As

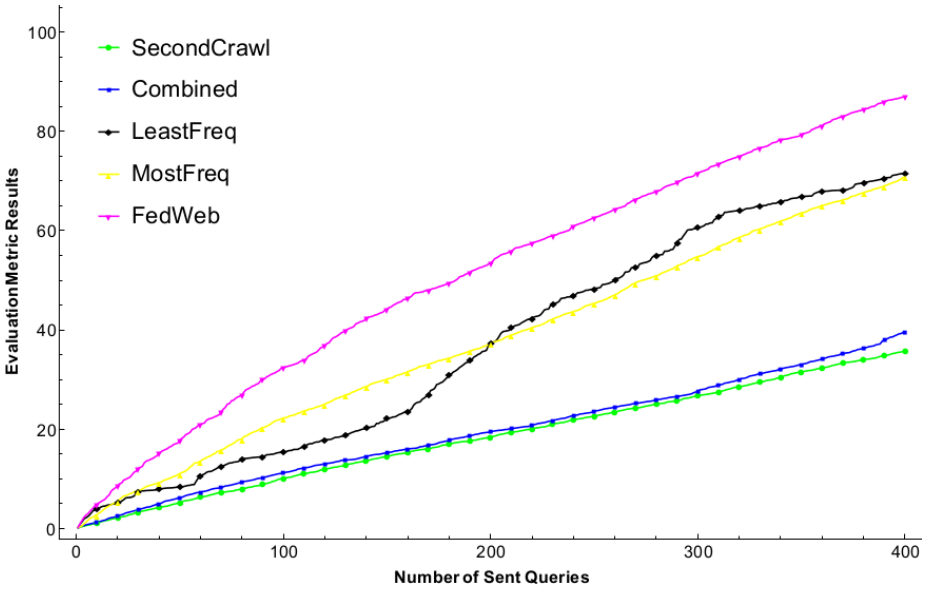


Figure 5.4: Performance of all the suggested approaches in returning changed documents for all test entities calculated through applying Equation 5.4

shown in Figure 5.4, FedWeb approach outperforms the other approaches by at least **20 percent** more changed documents retrieval. In following, we discuss the reasons behind this performance and mention our findings.

5.6 Analysis

To retrieve more changed documents, an approach is successful that can form queries that can return not only more changed documents but also fewer duplicates among these changed documents. To study this hypothesis, Figure 5.5, in 4 sub-figures, compares the generated samples by all the approaches.

Figure 5.5a shows the sizes of the generated samples by each one of the applied approaches to collect changed documents. In Figure 5.5a, FedWeb and MostFreq approaches produce big samples represented by points on the top area of the figure while the points representing LeastFreq. approach are more present in the bottom area of the figure which means producing small samples.

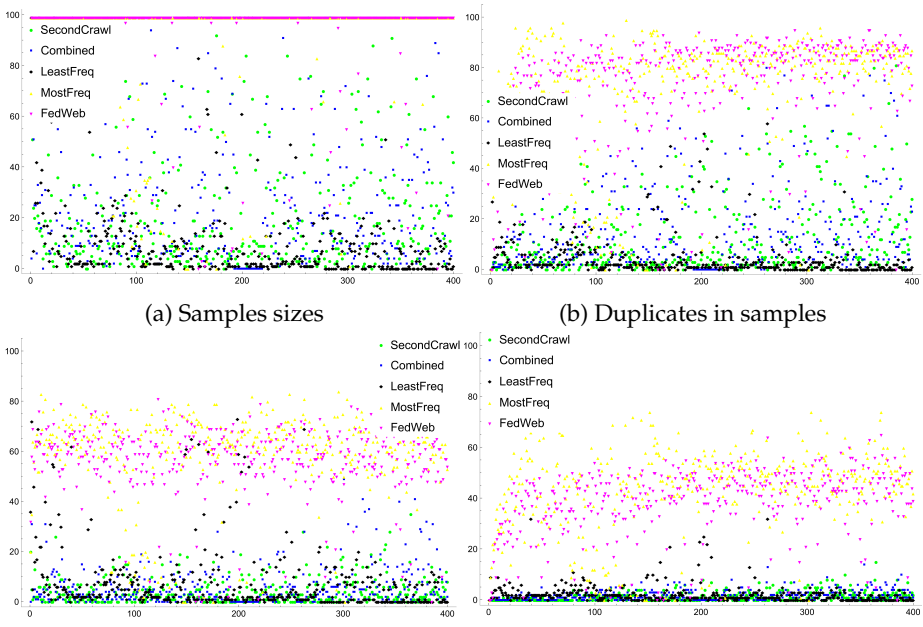
Figure 5.5b shows the number of duplicates among the generated samples. If the representing points of an approach are in the top area of Figure 5.5a and the bottom area of Figure 5.5b, the approach retrieves more unique documents. We repeat the same analysis on the samples but for the changed documents.

Figure 5.5c shows the number of changed documents retrieved in each sample. As it is shown in this figure, FedWeb and MostFreq approaches return the most number of changed documents in the samples. Figure 5.5d shows the number of duplicates among these changed documents. The maximum number of duplicates among changed documents is observed in generated samples by MostFreq technique. Although FedWeb produces more duplicates than LeastFreq and Combined methods, it performs better than MostFreq approach. If the representing points of an approach are in the top area of Figure 5.5c and the bottom area of Figure 5.5d, the approach retrieves more unique changed documents. Figures 5.5c and 5.5d show that FedWeb makes a better trade-off between bigger samples and fewer duplicates.

As shown in Figure 5.5, FedWeb is the best performing method in retrieving more unique documents considering Figures 5.5a and 5.5b. We also show in Figures 5.5c and 5.5d that FedWeb approach outperforms other approaches in returning not only more documents but also more unique changed documents.

In FedWeb technique, we send a representative set of words from changed content in FedWeb data collection. From the observed results in Figure 5.4, we can claim that these words are good representatives of changed content. In addition, Figure 5.5 shows that these selected terms have frequencies that suits the best to reach a trade-off between more documents and fewer duplicates.

As our samples are search engine results, if a page for a submitted query in our first harvest is no longer among the returned results for that query in our next harvest, there is a chance that page is not changed but not returned due to a change in ranking of results. To reduce the number of these pages and their effects in our results, while comparing pages, we consider not only the returned results for one query but also all returned documents in all the harvests. However, we noticed that if the first harvest for an entity does not reach a complete or high coverage on all the relevant documents, there is a higher chance of having such documents that are not changed but just absent in previous harvests. For such entities, approaches like LeastFreq. lead to better performance. In LeastFreq. approach, we submit queries formed by adding the least frequent terms among the already retrieved documents to a seed query. As these terms are less frequent, they return documents less likely to be covered by other approaches in previous harvests. From the results for each entity, we also noticed that change and its rate are subjective to each topic and domain. Although we



(a) Samples sizes

(b) Duplicates in samples

(c) New documents in samples

(d) Duplicates among changed documents

Figure 5.5: Performance of different approaches for "Fireworks Disasters" entity

noticed differences among the number of retrieved changed documents for different entities in the test set, FedWeb approach was always the top performing method in returning more unique changed documents. As another finding of this work, we noticed a big change in the top-100 returned results by Google. In SecondCrawl, we submitted the exact queries to Google and faced, in average, around 50 percent difference among the returned results in a period of two weeks.

5.7 Conclusion and future work

The main goal of the study in this chapter was to find efficient approaches to monitor web data for a topic of interest over time. In an ever-changing environment, devising efficient methods that can detect and retrieve changed web data for a given topic, from search engines imposing access limitations,

is a challenging task. To address this challenge, we proposed four different approaches to efficiently harvest changed documents matching a given entity by querying a web search engine. Among these approaches, FedWeb approach outperformed the others. FedWeb, which is based on a set of representative terms of changed documents in FedWeb data collection, produced more unique changed documents while tested on the tested search engine and entities. This approach performed the best, with at least 20 percent difference from the other tested approaches, in returning more unique changed documents with the same number of queries.

In this work, we also analyzed FedWeb collection to study change rates of different search engines and categories. We showed that FedWeb collection changed in average 40 percent across different websites. We also noticed that this change is highly subjective to domains and categories and varies from domain to domain. Our findings correlate with the results in previous literature studies where web data change rate is subjective to domains and topics [31,137] and showing the web is dynamic and changing rapidly. This change rate information is important for choosing the best time to re-harvest data sources to get new information for a given topic.

In general, the results of this investigation show that monitoring web data in focused deep web harvesting can highly benefit from adapting query generation mechanisms to detect and retrieve changed content efficiently. We showed that instead of re-running harvesting processes, with a well-designed strategy, more changed documents can be retrieved with the same costs which is important in an environment of access limitations and limited resources.

Future work In this work, we investigated the effects of the frequency of words, presence of a word in retrieved documents and representative terms of changed documents in selecting the terms to form the best next query to submit. In addition to these factors, we consider a number of issues such as dependency of a query candidate to all previously submitted queries or further analysis of returned results by applying techniques such as entity extraction, entity disambiguation and text parsing, as important factors that need to be studied as future work to reach more efficient change detection techniques. As another future work, while we chose FedWeb collection which focuses on search engines in this work, we can consider ClueWeb or Common Crawl data collections. By analyzing two different versions of a large web crawl, we can either assign weights to websites representing their corresponding numbers of changed pages or train a classifier for terms and changed documents. Either

through these weights or the classifier, we can select terms to expand a given seed query to form queries that can predict changes more accurately and efficiently.

CHAPTER 6

DESIGNING A GENERAL DEEP WEB HARVESTER BY HARVESTABILITY FACTOR

What are the key features to consider while designing and developing an access approach that can be applied to a wide range of websites, domains and tasks? How can we compare the performances of different web harvesters? How can we measure to which extent a website can be harvested? What is harvestability factor and how can it help to answer these questions?

This chapter is based on [82, 85].

To make deep web data accessible, harvesters have a crucial role. Targeting different domains and websites enhances the need of a general-purpose harvester that can be applied to different settings and situations. To develop such a harvester, numerous issues should be considered. To have all the influential features in one big picture, a new concept called harvestability factor (HF), is introduced in this chapter. HF is defined as an attribute of a website (HF_W) or a harvester (HF_H) representing the extent to which a website can be harvested or a harvester can harvest. The comprising features of these factors are features of different websites or harvesters. These features are gathered from literature or introduced through our experiments. In addition to enabling designers of evaluating where they products stand from the harvesting perspective, HF can

act as a framework for designing harvesters. Designers can define a list of features and prioritize the implementations of those features. To validate the effectiveness of HF in practice, we show how the features of HF can be applied in categorizing deep websites and how this is useful in designing a harvester. To validate $\mathtt{HF}_H$ as an evaluation metric, $\mathtt{HF}_H$ is calculated for *HarvestED*, our developed harvester, and *import.io*, a commercial harvester. The results show our developed harvester is more successful for the targeted test websites by scoring 14.8 out of 15.

6.1 INTRODUCTION

Through targeting different domains and websites in Chapters 3, 4 and 5, we noticed the importance of a general-purpose harvester that can be applied to different settings and situations. To develop such a harvester, a number of issues like business domain, targeted websites and harvesting goals should be considered. Different business domains and goals can demand diverse requirements from deep web access approaches. In some domains, a few big databases are the main sources of data and in others, data is scattered through many websites. The latter makes it more desirable to have an approach with no need of extra configuration or at least, a minimal configuration effort for each website. The goal of a harvesting task is also important [71]. If the goal is to extract all data and a harvester downloads the data partially, the harvesting task is not considered successful. However, this can be a success story if the goal is just to obtain a representative set of data [71]. In addition to the targeted domain and the harvesting goal, features of deep websites have great impacts on a deep web access approach. Different website features, from graphical interface to back-end designing and developing techniques, play an important role. If a website is written in Flash, as a Java applet or a simple HTML page, it makes a big difference for designing an access approach [5]. Without a well-defined list of features affecting harvesting tasks, having a general deep web access approach seems far from reach.

Contributions As our main contribution in this chapter, a new concept called *harvestability factor* (HF) is introduced. This concept enables designers of websites and harvesters to evaluate where their products stand from a harvesting point of view. Through this concept, we also put all the important features in harvesting deep websites in one big picture. This overall view provides a guideline for designing general deep web harvesters trying to cover all the

features from different aspects. Some of these factors are mentioned in the literature and the others are discovered through our experiments. Having all these features in one big picture, we also evaluate their importance in harvesting processes. This helps to have a better strategy to implement a harvester. Defining the importance of each feature helps to prioritize the implementations of features.

Outlook In Section 6.2, harvestability factor (HF) is introduced. After discussing related work in Section 6.3, Section 6.4 introduces the features of $\mathtt{HF}_W$ which are also applied for categorizing deep websites. In this section, all the features of deep websites that affect a harvesting process are introduced and deep websites are categorized accordingly. In Section 6.5, the features of HF for a harvester are defined. All the general and detailed requirements for designing a general-purpose deep web harvester are also discussed. The different approaches applied in the literature to meet these requirements are also explored. Having mentioned all the necessary requirements in Section 6.6, as examples of a deep web harvester, our designed harvester *HarestED* and a commercial harvester *import.io* are discussed. With using these example harvesters, HF as a design framework and HF as an evaluation metric are validated. Finally, Section 6.7 draws conclusions and suggests future work.

6.2 HARVESTABILITY FACTOR

To formalize all issues affecting a deep web data access approach, a new concept called *harvestability factor* (HF) is introduced in this chapter. Although in a harvesting process, the roles of both harvester and website are intertwined, separate definitions are required by designers of websites and harvesters for better understanding of harvesting processes. Hence, HF is defined as an attribute of a website or a harvester representing the extent to which, consecutively, a website can be harvested ($\mathtt{HF}_W$) or a harvester can harvest ($\mathtt{HF}_H$).

The HF of a given harvester h is calculated through Equation 6.1. In this equation, we differentiate between a general feature (gf) and a website feature (f). General features are discussed in more detail in Section 6.5. For general features, we include the harvester performance for each feature while for website features, we multiply the harvester performance by the importance of a feature. In Equation 6.1, the importance of a feature is shown by $\mathtt{Cr}_f$ and $\mathtt{Co}_f$. $\mathtt{Cr}_f$ shows how critical a feature is for a harvesting process and how much information you can still get from web pages if your harvester doesn't support

that feature. $\mathtt{Co}_f$ represents how often the feature f is used in the targeted domain. Harvester performance h_p is represented through harvester failure $\mathtt{Fa}$ and calculated through $\mathtt{Fa} = 1 - h_p$. $\mathtt{Fa}$ shows how well a harvester can handle a feature. For example, a given harvester can detect *XPaths*[1] for more than 50 percent of the cases correctly. However, this 50 percent failure leads to a data loss of 75 percent. Therefore, for this feature, the $\mathtt{Fa}$ of the harvester equals 50 percent while $\mathtt{Cr}_{\mathtt{xpathDetection}}$ is 75 percent. In Equation 6.1, n is the number of website features and k represents the number of general features.

$$\mathtt{HF}_H(h) = \sum_{i=1}^{n}(1 - (h_{\mathtt{Fa}_{f_i}} \times \mathtt{Cr}_{f_i} \times \mathtt{Co}_{f_i})) + \sum_{j=1}^{k}(h_{p_{\mathtt{gf}_j}}) \tag{6.1}$$

In Equation 6.2, $\mathtt{HF}_W$ is defined for a website considering the discussed features in Section 6.4. In this equation, given a website w, the averaged performance of all the tested harvesters for each feature of w is multiplied by the importance of that feature. n is the number of features and m is the number of considered harvesters. In this equation, $w_{p_{f_i}}$ represents the absence or presence of a feature in a given website.

$$\mathtt{HF}_W(w) = \sum_{i=1}^{n}(1 - (w_{p_{f_i}} \times (\frac{1}{m}\sum_{j=1}^{m}(h_{j_{\mathtt{Fa}_{f_i}}})) \times \mathtt{Cr}_{f_i})) \tag{6.2}$$

Assigning accurate values to the mentioned weights and features in these two equations is beyond the scope of this chapter and considered as future work. However, in Section (6.6), by using simple methods to assign values to these parameters, it is shown how these equations can help to evaluate harvesters and websites. In this chapter, we try to cover all aspects of the introduced $\mathtt{HF}$, such as business domain, harvesting goal, harvesters features and websites features, to suggest a design guideline for a harvester.

6.3 Related work

In this chapter, we target two issues regarding $\mathtt{HF}$. First, we study $\mathtt{HF}$ as a harvester design framework and secondly, we investigate $\mathtt{HF}$ as an evaluation metric for websites and harvesters. Since the introduction of the deep web, there have been several attempts to give access to this part of the web and

[1] $\mathtt{XPath}$ uses path expressions to navigate through elements and attributes and select nodes or node-sets in an XML document.

improve the existing approaches [14, 74, 91, 105, 116, 123, 136]. In all these approaches, the focus is on harvesters rather than websites. They try to improve the performance of a harvester by applying new techniques. Although this is essential but it is not enough. In this work, we believe that improving efficiency, scalability, robustness and other requirements of harvesters is not possible without having all the affecting factors in one big picture. Introducing HF is the first step in this direction. HF helps to study not only harvesters but also a target domain and the features of websites in that domain while designing a harvester.

Evaluate and compare harvesters Studies in the literature about comparing and analyzing web harvesting tools [72, 76] mainly focus on a limited number of aspects such as capability in dealing with different data formats, capability to record the extracted data, user friendliness, price in market, export formats, ability to manage the anonymous scraping and multithreading. However, in this study, in addition to these features, a more detailed and comprehensive set of features are introduced. Despite others, we provide a mechanism to assign a number to each harvester considering a wide range of general and detailed features.

6.4 FEATURES OF THE HARVESTABILITY FACTOR OF A WEBSITE

In this section, the features of websites that can influence harvesting processes are studied. The role of each feature in defining $\mathtt{HF}_W$ is also mentioned. These features are also applied for categorizing deep websites from the harvesting perspective.

6.4.1 Web development techniques

A number of applied techniques in developing and designing websites and web pages create challenges for harvesters. These techniques are usually applied to add interactivity to web pages and improve site navigation. In following, a list of such techniques is presented.

1- Embedded scripting languages in HTML pages

Embedded scripts in HTML pages can change content of a page based on changes in page environment such as user action or time. Embedded scripts in HTML pages can make content in layers either shown or hidden based on a user action. The embedded scripts can even build HTTP requests to fill out and submit a form dynamically. Client-side scripts can manage HTML layers, perform redirections, dynamically generate navigations like pop-up menus and create hidden anchors [5, 49]. These changes may prevent harvesters to analyze a page as it is shown to users.

2- Session management mechanisms

A session management mechanism in a server keeps track of transactions with users. Based on the transactions history of a client and information on client resources, a server can change the services that it provides to users. For harvesters, in later access to documents or in distributed crawling, as user environment may change or the session may expire over time, session management mechanism can cause problems for harvesters [5].

3- Complex URL redirections

URLs are redirected for a number of different reasons such as handling similar or moved domains, manipulating search engines or visitors and shortening URLs. This means different responses, which might result in different pages, are given to a browser request. These redirections happen automatically or manually. Automatic redirections are initiated on either server-side or client-side. It is easier for harvesters to deal with redirections handled on server-side unless it is a never ending redirection loop (not loading any page) or a redirect chain that may take longer than expected to reach a final page [5]. Handling redirections initiated by embedded scripts in page content is a completely different story. Refresh meta tag in HTML, JavaScript redirections and frame redirections are examples of these user-side redirections.

4- Applets or Flash code

If Flash or Java applet are used for designing all pages in a website, it is almost impossible for harvesters to access its content without running an expensive analysis. Nowadays, web designers avoid these practices to make sure their sites are on good terms with crawlers. If only the welcoming page is designed

by Flash or Java applet, it becomes easier for harvesters. Of course, application of Flash in advertisements has no effect and is ignored.

5- Frames

Existence of *HTML frames* in pages may also create difficulties for harvesting processes. Detecting the right frame that contains the page content in a multi-frame page is one of the problems.

6- HTML coding practices

For harvesters relying on tags, attributes and presentation features, HTML code practices are important. Having bad-written HTML code (e.g. unclosed tags) causes problems in analyzing the HTML tree of a page and therefore, failure of harvesters in extracting data. Lacking well-defined IDs, classes and other explanatory attributes for items creates difficulties for harvesters and makes them prone to mistakes. Being consistent in coding practices for all pages and data items is also important. For example, if IDs are used for items, it should be the case for all of them or at least a defined set of items (e.g. different categories). In some cases, items from the same category, even with the same presentation template, have small differences in HTML codes behind them. This inconsistency can mislead harvesters.

6.4.2 Website policies

1- Search policies

Query interfaces Web interfaces are classified as keyword-based, form-like, browsing or a combination of them [144]. Each one of these interfaces imposes a different set of requirements for harvesters. For example, in a form-like search interface, information on attribute-value bindings and accessing predefined lists of values for attributes are helpful for harvesters in selecting queries to send to search engines. Detecting interfaces and recognizing different features of web forms are also helpful for harvesters. For example, if a query interface enables searching by industry domain, region or time, a harvester can use these features to perform more efficiently.

Indexing policies For harvesters targeting websites with search possibility, knowing about the indexing policies is important. For example, if stop words

are indexed in a website, sending a stop-word query is one of the most reliable options to have results. Also, if there is no limitation on browsing through search results, sending only one stop-word results in a high coverage. In addition to indexing policies regarding stop words, it is important to know which parts of data are indexed. For example, if only page titles are indexed, selecting next queries from data in all sections of pages may return no results. In Chapters 3, 4 and 5, where we chose terms to form next queries from already retrieved results, we experienced the effects of such policies.

Search queries and algorithms Websites do not necessarily follow similar principles in replying to a submitted query. In some cases, stop words are removed from search queries, query phrases are treated in different ways (considered as an AND phrase or an OR phrase) or the number of returned results shown to users is different. There may be even differences on additional information provided in reply to a query such as statistics on search results and the number of found relevant documents. There are also websites posing limitations on the number of queries a client can send. In Chapters 4 and 5, we extensively discussed the results of such limitations on harvesters.

Navigation As a common practice in designing websites, a query is sent, search results are displayed and by following each one of those returned results, a detailed page is presented. However, there are websites that return a list of categories for a submitted query. Following each one of those categories can end up in another subcategory. This difference in navigation from search results to detailed pages makes it difficult for a harvester to realize that a returned result is a category instead of a detailed page.

2- Security, privacy and legal policies

Answering this question is one of the first steps in a harvesting process: "is it legal to access data, store it and present it to users?". It is also important to check if credentials are required by a website to access its content. To follow the privacy policies, it is also important to consider the terms of services of a website. Some websites use *Robots Exclusion Protocol* giving instructions to web robots in a file named *Robots.txt*. In case of the existence of such a file, its instructions should be considered. Not all websites welcome bots (harvesters or crawlers) with open arms. Through traffic monitoring, bot identity declaration or real person declaration techniques like a CAPTCHA, websites can detect bots and use various measures to stop or slow them down. Blocking an

IP address, disabling a web service API, commercial anti-bot services or using application firewalls are some of these measures. It is also important to note other privacy policies of a website like policy on disclosing aggregate information for analytic purposes by owners of websites.

6.4.3 Data and content

1- Type and format of residing data in data sources

Deep websites are categorized into two groups based on their content [71]: structured data (e.g. almost all shopping websites with products as entities, movies sites, jobs listings, etc.) and unstructured data (e.g. textual content, GPS traces, etc.). Pages from these groups have different characteristics that can affect the performacne of a harvester. This differentiation defines which harvesting techniques (discussed in Section 6.5.2) are suitable for harvesting a page from each one of these groups. For text, applying a natural language processing technique is a better option while for a structured page, HTML-based harvesters end up in better results. It is important to know about different data file formats (e.g. PDF, image or video files). Different data formats need different handlers to download them. If this is not considered by harvesters, they fail in extracting data.

2- Data layout

To harvest data from structured pages, different data presentations affect harvesters that depend on presentation features of data items. Different data items in a website can be presented in different ways. For example, shoes are presented differently from books. Even data items of a same category can be presented differently based on their features (history books vs. fiction books). If these data presentation differences are not known to a harvester, it uses the same algorithm to extract all data. This results in extracting none or undesired information. Structural variations on data presentation must be tolerated by harvesters and treated accordingly.

3- Data type formats

For harvesters relying on ontologies and text patterns for extracting data from detailed pages, it is important to investigate how they can affect a harvesting process. Committing to one ontology and following same data patterns for same concepts make the configuration and design of a harvester much easier.

For example, if all the dates mentioned in a website follow the "dd-mm-yyyy" format, a harvester can easily be configured to extract all the dates. However, for example, if address format is not the same for all the addresses in a website, it adds complications to the harvester configuration and might result in no data or undesired information extraction.

4- Information of a data item is scattered in different pages

Some search engines return the relevant information to a submitted query not in one page but scattered among a number of pages. In these cases, the general information about the query is presented in one page. However, for more detailed information, users should visit some other links accessible (only) through this detailed page (you need to go to the detailed page and then browse through the tabs or links to access the information you want). Finding these links and extracting information from them is a challenging task for harvesters.

5- Providing semantic annotations (metadata)

A page may include metadata or semantic markups and annotations. Annotations may be embedded in pages or organized into a semantic layer [53] stored and managed separately from web pages. Extracting data schema and retrieving instructions from this layer before scraping pages can help harvesters to perform more accurately and efficiently in extracting data.

6- Website content language

For harvesters that perform based on parsing the content of web pages like using data patterns, knowing about the language of a website and being capable of dealing with that language are necessary. Dealing with Chinese language needs different configurations than English or Farsi languages. Having different languages in the targeted websites also causes difficulties for harvesters.

6.5 Features of the harvestability factor of a harvester

As mentioned in Section 6.1, designing a deep web access approach is highly affected by business domains, websites and harvesting goals. In the previous section, we studied the features of websites as a set of features affecting HF.

The ability of a harvester in dealing with each one of these mentioned website features is included in defining HF for a harvester. In addition to this set of features, there are a number of general requirements that should be met by a harvester. This set of general requirements is introduced in Section 6.5.1. The performance of a harvester for these two sets of features is considered to define the HF for a harvester. To evaluate the harvester performance for each one of these features, we need to have more information about the applied techniques in harvesters. Therefore, Section 6.5.2 is dedicated to define these methods and techniques.

6.5.1 General features

Every general-purpose harvester, regardless of its goal and its targeted domain and websites, should meet a set of requirements that are important in all harvesting processes. Being *automatic* or running with minimal configuration is one of these requirements. Being *scalable* (applicable to many websites), *independent* (of business domain, technology, etc.), *efficient* (with the least possible number of queries, harvests the most possible amount of data), *easy-to-use* (configuration and settings should be easy for users) and *resilient-to-change* for both the website content and its presentation are other general requirements that should be considered in evaluating the performance of a harvester.

The *robustness* of a harvester is also important in a harvesting process. It enables harvesters of doing an uninterrupted harvest as they can detect and resolve issues like IP-based blocking and websites failures. A harvester should be capable of providing information or, if required, firm guaranties about the *amount of reached coverage* in harvesting the targeted websites. The size estimation of deep websites and also defining a stop condition for a harvesting process can help to achieve this capability, as discussed in Chapters 1 and 3.

For harvesters with the goal of monitoring the content of pages on the web, it is important to keep the harvested data up-to-date. This implies that harvesters should be capable of detecting new and deleted content on the web. The following section studies the applied methods in harvesters and how they affect the *HFs* of a harvester.

6.5.2 Harvesting techniques

To access data behind web forms, various harvesters are suggested in the literature [14, 74, 91, 105, 116, 123, 136]. The differences among these harvesters originate from different sources, from applied techniques to main harvesting

goals. In this section, we categorize harvesters based on the applied techniques and tools to meet the introduced requirements in Sections 6.5.1 and 6.4. This categorization helps to understand why a harvester is successful for one website and not for another. It also helps to evaluate the harvester performance for a website even before applying it in practice. If we know harvesters from a category fail in harvesting websites with a specific feature, we can predict the performance of a harvester from that category for a given website with the same feature.

This categorization is based on the proposed taxonomy by [91].

1. *HTML-based harvesters* HTML-based harvesters rely on a set of different features of the HTML code of a document [40, 75, 87, 135, 141]. To analyze the HTML structure of a page, the page is translated into a parsing tree. This translation can be performed by using browser controls (like browser controls of Internet Explorer) to parse web pages into *Data Object Model (DOM)* trees. Then, by running a number of predefined extraction rules on the tree, the data is extracted.

2. *NLP-based harvesters* NLP-based harvesters [55, 112, 118] apply different Natural Language Processing (NLP) techniques such as filtering, part-of-speech tagging and lexical semantic tagging for building relationships among sentences, their parts and phrases. From these extracted relationships, a number of extraction rules can be derived. These rules are based on syntactic and semantic constraints and help to identify the relevant information within a document.

3. *Machine learning based harvesters* These harvesters [90] rely on a given set of training examples to derive a number of extraction rules. In these techniques, rather than relying on linguistic constraints found in a web page, rules are based on features of the structure of pieces of data.

4. *Modeling-based harvesters* In modeling-based harvesters [2, 42, 125], a data model is defined. In this data model, a number of objects, their properties and relationships are defined. Based on this data model and its modeling primitives, points of interest are located in web pages.

5. *Ontology-based harvesters* In ontology-based harvesters [50], the extraction process is based on data and not the presentation structure. These harvesters need a specific domain ontology. Through domain ontologies, the relevant concepts to a particular topic or area of interest are defined

and made available for harvesters. Ontology-based harvesters use these ontologies to locate ontology's constants present in the page and to construct objects associated with them.

6. *Computer vision based harvesters* These harvesters use computer techniques from the domain of computer vision in addition to techniques from machine learning to analyze web pages. In these harvesters, the main goal is to identify and extract information from web pages by interpreting them visually as a human being does. Some of these approaches use also the visual features of deep web pages [100].

7. *Combined harvesters* Harvesters can be based on a combination of the previously mentioned categories. For example, an HTML-based harvester can apply machine learning techniques to increase its accuracy in extracting results.

6.6 HARVESTABILITY FACTOR VALIDATION

As mentioned in Section 6.1, `HF` can be used for evaluating both a harvester and a website. It was also discussed that this factor can be applied as a design framework. To validate these claims, we study the features of `HF` on a collection of deep websites as our test set. We also evaluate *HarvestED* [79] and *import.io* [75] harvesters on this test set. `HarvestED` is our developed harvester which stands for Harvest Entity related Documents and import.io is a successful commercial example for tools developed for scraping web data.

6.6.1 Test set

To create a test set for illustrating how deep websites can be categorized based on `HF` features and how this categorization is applied to design a general-purpose harvester, a number of websites are selected from the list of top-100 job vacancy websites [130]. To extend this test set, a number of Dutch job vacancy websites are also considered. For each of these websites, all the features of `HF` are studied. We examine the performances of our test harvesters on the websites from each one of these categories.

6.6.2 HarvestED

HarvestED is our developed harvester which is a combination of both model-based and HTML-based categories. To harvest a website by HarvestED, we need to define a website template including points of interest and their corresponding XPaths. HarvestED gets this configuration information from either a file or a database table and starts harvesting a given website through simulating the loading of pages in real web browsers. For each returned page, whether it is a results page or a detailed page, the points of interest are extracted by running XPath queries. The extracted data can be stored either in files or database tables. The configuration is limited to entering the template and XPaths for points of interest and there is no need to enter a data model for data storage.

These characteristics of HarvestED help to resolve the caused challenges by some features of websites mentioned in Section 6.4. For example, to enable our harvester to implement embedded scripts in HTML pages, we use techniques for automating browsers. We also consider HTML-based techniques for selecting the points of interest. These features also help a harvester to meet the mentioned general requirements in Section 6.5.1 like automation, scalability, independence, robustness, efficiency and being easy-to-use. For efficiency purposes, different query generation mechanisms are used to retrieve the maximum amount of data with the least possible number of posed queries. Domain-independence is also achieved through only using HTML-based techniques which also makes it language-independent.

6.6.3 Import.io

`Import.io` [75] is a successful example of developed tools for scraping web data. Import.io is a combination of machine learning and HTML-based categories. It analyzes HTML codes to detect lists on web pages and also uses XPath queries for extracting data from web pages. Through using machine learning techniques, import.io can handle small changes to a website automatically and continue to get data. However, it fails in dealing with a complete change in the design of a website. The crawler of import.io discovers sites by following links from page to page and does not consider other navigation methods for increasing the efficiency, regarding the number of submitted requests, or for focused web harvesting purposes. It does not work for authenticated APIs either. One of the advantages of import.io is its easy-to-use configuration user interface which makes it possible for users to easily configure and run the scraper.

6.6.4 Validation of HF as a website/harvester evaluator

In this part, we study the possibility of using HF to evaluate the harvestability of a website or a harvester. As mentioned in Section 6.2, assigning values to weights and features in the equation for $\texttt{HF}_W$ is beyond the scope of this chapter. However, to show how this is beneficial, we use a simple method to assign these numbers. All the values are assigned with probabilities. We assign the percentage of the occurrence of a feature in the test set as $\texttt{Co}_f$. The $\texttt{Fa}_f$ values are calculated through $\texttt{Fa}_f = 1 - \texttt{success\_rate}_f$ equation. We assign $\texttt{success\_rate}_f$ based on the capability of a harvester in resolving the problems caused by features of the tested websites. We also assign $\texttt{Cr}_f$ values based on our experience and the observed results from running the harvesters. $\texttt{Cr}_f$ values represent how influential a feature f is in a harvesting process. We assign them with four values, *Highly Critical (1)*, *Very Critical* (0.75), *Critical* (0.5), *Effective not Critical* (0.25) and *no effect* (0).

Having assigned all the present parameters for HF with values in Table 6.1, $\texttt{HF}_H$ can be calculated. This calculation is shown in Equation 6.3.

$$\begin{aligned}\texttt{HF}_H(\texttt{HarvestED}) = 8 &+ (1 - \frac{26}{100} \times \frac{50}{100} \times \frac{70}{100}) \\ &+ (1 - \frac{2}{100} \times \frac{75}{100} \times \frac{50}{100}) + (1 - \frac{2}{100} \times \frac{50}{100} \times \frac{80}{100}) \\ &+ (1 - \frac{14}{100} \times \frac{50}{100} \times \frac{10}{100}) + (1 - \frac{10}{100} \times \frac{50}{100} \times \frac{10}{100}) \\ &+ (1 - \frac{1}{100} \times \frac{75}{100} \times \frac{80}{100}) + (1 - \frac{24}{100} \times \frac{75}{100} \times \frac{20}{100}) \\ &= \mathbf{14.849} \end{aligned} \tag{6.3}$$

The best score is the result of having all the features equal to 1 which results in 15. HarvestED scored 14.84. This high number tells us HarvestED is successful for the targeted domain. The absence of eight features of the tested websites in this domain gives a big advantage to this harvester. In this part, we only show the $\texttt{HF}_H$ of HarvestED for a limited test set. With an averaged weight for each parameter in the $\texttt{HF}_H$ equation, the harvester performance can be tested generally.

As shown in Equation 6.4, we also calculated the $\texttt{HF}_H$ of import.io [75], a web data extraction tool. To assign weights to the performances of *import.io* for each feature of HF, we ran import.io on a number of sample websites, the documentation of import.io and also information required from email contacts with the developers. In case of no available information on a HF feature f,

Table 6.1: Assigning weights for HF features for the test set websites and harvesters

HF feature (f)	**$\mathtt{Co}_f, \mathtt{Cr}_f$**	**$\mathtt{HarvestED}_{\mathtt{Fa}_f}$**	**$\mathtt{import.io}_{\mathtt{Fa}_f}$**
Embedded script in HTML 6.4.1	%100 VeryCritical(%75)	Successful⇒ *%0*	Successful⇒ *%0*
Applet/Flash 6.4.1	%0, HighlyCritical (%100)	This feature is not included and harvester fails.⇒ *%100*	This feature is not included and harvester fails⇒ *%100*
Data layout6.4.3	%26 Critical(%50)	needs preconfiguration for different page templates⇒ *%70*	Detects lists but needs preconfiguration for different layouts⇒ *%50*
Navigation6.4.2	%2 VeryCritical(%75)	Successful(differentiates search result and detailed pages) ⇒ *%50*	not documented (considered successful) ⇒ %0
Multi-page data source 6.4.3	%2 Critical(%50)	needs preconfiguration⇒ *%80*	needs preconfiguration but easier through UI⇒ *%60*
Search policies 6.4.2	%14 Critical(%50)	Solved by different query generation mechanisms⇒ *%10*	not documented (considered successful) ⇒ %0
Indexing policies 6.4.2	%10 Critical(%50)	Harvester detects if stop words are indexed or not⇒ *%10*	not documented (considered successful) ⇒ %0
HTML coding practices6.4.1	%0(all persistent) Critical(%50)	problem as it is HTML-based⇒ *%80*	problem as it is HTML-based⇒ *%80*
Security privacy legal policies 6.4.2	%0 (no credentials or bot limitation) VeryCritical(%75)	deals with cookies ⇒ %50	follows limitation for bots ⇒ %50
URL redirection 6.4.1	%14 Critical(%50)	Successful ⇒ %0	Successful ⇒ %0
Residing Data 6.4.3	%10 Critical(%50)	Successful ⇒ %0	Works with XPath RegEX ⇒ %0
Session management 6.4.1	%2 VeryCritical(%75)	deals with cookies ⇒ %0	not documented (considered successful) ⇒ %0
Query interface type 6.4.2	%100(all enable text search or browsing) VeryCritical(%75)	Successful ⇒ %0	Only browsing ⇒ %70
Persistent data patterns 6.4.3	%24 VeryCritical(%75)	Successful if defined data layout ⇒ %20	Successful if defined data layout ⇒ %20
Multi-frames 6.4.1	%1 VeryCritical(%75)	Fails to detect main frame ⇒ %80	Fails to detect main frame ⇒ %80

we assigned the success rate for that feature as 1 ($\texttt{success\_rate}_f = 1$) and therefore, $\texttt{Fa}_f = 0$. The assigned weights are shown in Table 6.1. Based on these weights, we calculated $\texttt{HF}_H$ for import.io through Equation 6.4.

$$\begin{aligned}\texttt{HF}_H(\texttt{import.io}) = 10 &+ (1 - \frac{26}{100} \times \frac{50}{100} \times \frac{50}{100}) + (1 - \frac{2}{100} \times \frac{50}{100} \times \frac{60}{100}) \\ &+ (1 - \frac{100}{100} \times \frac{75}{100} \times \frac{70}{100}) + (1 - \frac{1}{100} \times \frac{75}{100} \times \frac{80}{100}) \\ &+ (1 - \frac{24}{100} \times \frac{75}{100} \times \frac{20}{100}) = \mathbf{14.362} \end{aligned} \tag{6.4}$$

Import.io scored 14.36 out of 15. This high number shows that import.io is also successful for the targeted domain assuming the harvester is completely successful for undocumented features. Although it is a high number, it shows that HarvestED is a better choice for extracting information from the domain of job vacancies.

6.6.5 Validation of HF as a framework

In this study, it is shown how websites are categorized by applying the features of HF and how this can guide the design and implementation of a harvester. Having studied a set of deep websites and prioritizing their features, by applying HarvestED and import.io on this set of websites, we showed how these features are effective on harvesting processes in practice.

The obtained results from studying the features of HF among the given websites in the test set reveal the common characteristics of job vacancy websites. If we assume the selected websites represent the job vacancies domain, the results can guide the design of a harvester by emphasizing on the features of HF faced more frequently. As can be seen from Table 6.1, embedded scripts, query interfaces, data layouts and changing data patterns occur more frequently and need further attention.

One of the design solutions to overcome the caused challenges by embedded scripting languages in HTML pages is to rely on browser simulation in a harvester. This is a valid solution unless a page changes by a user action, a change in user browser or passing time. In these cases, to harvest the page as it is exactly presented to users, we need to simulate the causes of changes in page content (like user actions). As this type of scripts was not faced in our test collection, in the design of our harvester, we considered only the first solution

and postponed the implementation of the second solution as a strategic design decision based on our observations.

The second most common HF feature in the test set is to detect query interfaces. This emphasizes the importance of developing harvesters that can detect the templates of websites and query search engines to harvest job vacancies. It can also be seen from Table 6.1 that dealing with query interfaces is very critical for harvesters. Therefore, it should be prioritized to be implemented in early stages of developing a harvester for extracting job vacancies.

The other commonly faced feature is different data layouts. This can be resolved if a harvester allows defining different page templates for each website. However, if there are many page templates used for showing data, this may require much effort during the configuration phase. In HTML-based approaches, data can also be extracted based on the content. Therefore, if the data patterns are consistent, a high quality data extraction is still possible.

Among the websites in the test set, 15 percent of websites have limitations on the number of search results a user can view. To resolve this problem, as discussed thoroughly in Chapter 4, different query generation mechanisms can be applied for efficient harvesting of deep websites. A harvester can also detect if stop words are indexed or not and send the next queries accordingly. These meet two other common HF features mentioned in Table 6.1.

During the experiments, it was also observed that users were asked to enable the cookies for a given website. This technique is becoming more frequently used by web developers. Therefore, harvesters should accordingly be able to recognize and resolve it. To resolve other session management techniques, keeping the session information and tracking the navigation path to a page are useful. However, with a less than 10 percent occurrence rate among the test cases, this has a low priority to be implemented in a harvester for extracting job vacancies.

An indirect search navigation in a website is another HF feature that is not common among job vacancy websites but is very critical to harvesting processes. One of the solutions to resolve the navigation feature of HF is by using page templates. Provided that there are only two types of page templates (search results page and detailed page templates) in a website, HarvestED can distinguish these two types and act accordingly.

As can be seen from Table 6.1, no websites in the test set were found for some HF features. This is due to the specifications of the test domain. For example, the application of techniques like Java applet or Flash are seen more frequently in domains like graphics or music industries and not so often in the job vacancy domain. The same applies to requiring credentials to view

job vacancies which is unusual in business models of these companies. It is also worth mentioning that detecting some of the features of HF for a website is time-consuming and sometimes hard. Persistent coding practices is one of those features. It is time-consuming to verify that a website does not follow a persistent coding paradigm unless you face an exception.

6.7 CONCLUSION AND FUTURE WORK

As discussed in Section 6.6, the features of the introduced harvestability factor can categorize websites based on their important features for a harvesting process. This enables owners and designers of deep websites to evaluate where their products stand from a harvesting point of view. This helps them to decide about what measures to take to follow their policies, whether it is increasing access or limiting it. They can also make decisions for the design of a website by knowing what is common in a target domain, hence what can be expected of the capabilities of harvesters used for that domain.

For harvester designers, HF acts not only as an evaluation metric of the performance of a harvester for different websites, it is also a framework for designing deep web harvesters. HF provides designers with a thorough list of requirements they should meet and also helps them to prioritize the features to be addressed and included in a harvester. While focusing on a target domain, a designer can prioritize development of a harvester by looking at common and critical features of websites.

Considering HF as a comparison metric for different deep web harvesters is another advantage of this introduced concept. To show how this can be applied, we calculated the equations for our own developed harvester and also a commercial scraper on a predefined set of job vacancy websites. To enable the calculations through the equations, we applied simple methods to assign values to weights in the equations. The importance of each feature was judged in a combination of our experience and expertise with the frequency of occurrence of that feature among the test set websites. Having more than half of the features absent among the websites gave an advantage to our developed harvester, HarvestED, to get the high score of 14.8. Of course, this shows that our harvester is very successful for this set.

To judge the performance of a harvester for a bigger or a different domain, we need to assign values that reflect the importance of HF features in the other domains and apply them in the equation. In general, the availability of averaged values for the importance of HF features will make it possible to evaluate

harvesters against a bigger set of websites. This also helps developers to decide what to include in a harvester and predict its outcome in the form of an accurate number, even before running the harvester on a targeted website. Calculating and assigning these average values is considered as future work.

As another future work, we need to study how to assign more accurate values automatically. This implies classifying websites based on HF features and evaluating each feature's importance automatically. As another future work, we aim at using HF to guide us in developing a more general deep web harvester. Using the studies performed in this chapter and extending them to a bigger test set will help us to decide about the features that a deep web harvester should include and prioritize their developments.

CHAPTER 7

CONCLUSION

In this thesis, we investigated the path towards a focused web harvesting approach that can automatically and efficiently query websites, navigate through results, download data, store it and track data changes over time. Such an approach can assist users in accessing a complete collection of relevant data to their topics of interest and monitor it over time. To reach this end, we explored all the obstacles and focused our attention on four of these challenges. We revisit the research questions in Section 7.1, suggest directions for future work on deep web content monitoring in Section 7.2 and conclude the thesis in Section 7.3.

7.1 RESEARCH QUESTIONS REVISITED

Below, we discuss our findings for each of the mentioned research questions in Chapter 1.

7.1.1 Complete coverage in focused web harvesting

In focused web harvesting, the goal is to harvest all the relevant information to a given topic from the web. Even with a fully automatic general harvester, achieving a thorough data coverage on a given topic is a challenging task. Some search engines, in both surface web and deep web, restrict the number of requests from a user or limit the number of returned results presented to

him. These limitations make it harder for harvesting all the related documents to a given topic. Even without these restrictions, harvesters should adhere to politeness policies which limits the number of submitted requests [113, 119]. Therefore, we investigated methods that could efficiently collect information for a given topic (i.e. with as few requests as possible) even with these restrictions. This question was formulated as *RQ 1*.

RQ 1: How can we improve data coverage of harvesters given the imposed limitations by search engines, limited resources of harvesters and adherence to politeness policy?

To address this challenge, in Chapter 4, we proposed a new approach, called *Comb.LB.FB*, that automatically collected information related to a given query from a search engine imposing both *#ResultsLimited* and *#RequestsLimited* limitations. This approach minimized the number of queries that needed to be sent by analyzing the previously retrieved results (Feedback-based) and combining this analyzed information with information from a large external corpus (List-based). Unlike existing approaches that focus on specifications of web databases or form inputs to efficiently harvest a deep website, we relied only on keyword search and query generation mechanisms in retrieving relevant documents to given topics. In Chapter 4, we experimented with several query generation strategies such as *LB-FixedFreq.*, *FB-MostFreq.*, *FB-LeastFreq.*, *FB-LeastFromLast* and *Comb.LB-FB*. We tested these approaches on real sources on the web. Among these strategies, *Comb.LB.FB* outperformed the best of the other approaches in data coverage for all the tested entities submitting the same number of queries.

7.1.2 Size estimation

Harvesting processes continue, while they have not achieved their goals, until they face query submission limitations posed by search engines or consume all their allocated resources, without any understanding of what percentage of the targeted collection is harvested. To avoid this undesirable situation, we need to know the status of a harvesting process, regarding the amount of downloaded data, as the process goes on. This helps to make a trade-off among the resources being consumed, limitations and the percentage of a website content being harvested. The size information of the targeted website helps to choose the best time to stop a harvesting process avoiding unnecessary consumption of resources. This is especially important in harvesting a non-cooperative website that hides the true size of its data. For these websites, the harvester cannot

decide if the website is fully harvested. Therefore, it is important to find methods that disclose this hidden size information of a website. To do so, the second question was formulated as *RQ 2*.

RQ 2: To improve the efficiency of a harvester, how can we estimate the size of a targeted website?

To address this question, in Chapter 3, we extensively reviewed the suggested approaches in the literature and categorized them based on their required information and also their applied methods in dealing with *query and ranking biases*. From each category, the discussed approaches including *MCR, MCR-Reg, G-MCR, CH, CH-Regression, Bar-Yossef* et al. and *Mhr* were implemented and compared in a real environment. We proposed **7** different methods such as *M-MCR, M-MCR-Reg, M-CH-1, M-CH-1-Reg, M-CH-2, M-CH-2-Reg* and *M-Bar-Yossef* based on modifications of the existing approaches. Our proposed approaches were also evaluated and compared. Among all these tested approaches, the proposed M-Bar-Yossef performed the best on our test set by a difference ranging from 35 to 65 percent from all the other tested approaches. In M-Bar-Yossef that applied a pool-based method, we suggested an intelligent pool selection technique. This pool selection technique allowed finding the best matching pool for a collection without any extra costs and improved the method by Bar-Yossef *et al.* [11] by selecting better pools.

7.1.3 Efficient change detection in focused web harvesting

In focused web harvesting, that aims at harvesting all relevant information to a given query, the fast evolving web creates challenges for keeping a harvested collection up-to-date. If we assume that all the relevant information to a given topic is already collected, keeping this collection up-to-date is a challenging task. Considering the costly process of focused web harvesting, it is important to devise methods to facilitate efficient re-harvesting processes that find only the new information for a given topic. This was formulated as *RQ 3*.

RQ 3: Given the ever-changing nature of the web, how can we keep the collection of the related documents to given topics of interest up-to-date and monitor it over time?

To answer this question, in Chapter 5, we discussed methods to detect data changes on the web for a given query over time. In this chapter, we introduced a new approach called *FedWeb* that efficiently harvested that changed

documents matching a given entity. In FedWeb approach, a list of representative words of changed content was generated by analyzing an external corpus. This corpus included harvested results for more than 300,000 queries on 150 web search engines in two different points in time. The FedWeb approach combined the information from this list with analyzing the content of the retrieved results of previous query submissions to a search engine. In Chapter 5, we proposed *LeastFreq*, *MostFreq*, *Combined* and *FedWeb* approaches to efficiently harvest the changed content in a deep website and tested these strategies on real sources on the web. Among these approaches, the FedWeb approach resulted, on average, 20 percent better change detection of the best of the other three approaches.

7.1.4 Designing a general deep web harvester

One of the important steps in designing a web harvester is to decide which features to include. We need to know the important issues in a harvesting task and how they influence the performance of a harvester. To have a general-purpose harvester which can be applied to different websites, domains and settings, we need to know all the influential elements on harvesting tasks and minimize the need for site specific configurations. A design framework and guideline are helpful in implementing such a harvester. These issues were formulated as *RQ 4*.

RQ 4: What are the features to consider while designing and developing a general access approach that can be applied to a wide range of websites, domains and tasks? How can we prioritize implementations of these features?

This challenge was addressed in Chapter 6 by investigating all the important features in designing and developing a general-purpose deep web access approach. Chapter 6 included not only a thorough literature study but also our experiments and learned lessons. We gathered all the influential elements on all the aspects of a harvesting process, discussed and categorized them. We also described different applied techniques in harvesters to address difficulties resulting from any of these elements. This information provided a detailed overview on designing deep web harvesters.

To have a completely automated web harvester, we needed to precisely define where the current harvesters stand and on what dimensions they needed to be improved. Due to the existing diversity of websites, domains, harvesting techniques and tasks, comparing harvesters by only the amount of the collected

data did not reveal all information on their performances. We introduced a formula to assign values to websites and harvesters based on the extension to which they could be harvested or harvest (harvestability). We applied this formula to HarvestED (Harvest Entity related Documents) [79], our open source harvester implementation and import.io, a business example of tools developed for scraping web data, and showed the possibility of such evaluation for harvesters. The results of these calculations showed which features are important while targeting the domain of job vacancies and which features needed to be added to the harvester to increase its performance.

7.2 FUTURE RESEARCH DIRECTIONS

In this book, we covered several aspects of implementing a deep web content monitoring approach, ranging from designing a general-purpose harvester to addressing issues in efficient focused web harvesting and change detection. At the end of each chapter, we extensively discussed future work. Here, we present a summary of the discussed points and add general discussions.

Throughout this thesis, we made tangible improvements towards having a completely automated deep web content monitoring approach. To move further forward, all depicted challenges in Figure 2.2 such as discovering relevant websites, understanding their interfaces, navigating through returned results and extracting data, storing data, monitoring data and presenting it to users should be addressed. Among these challenges, we focused on efficient focused web harvesting (through examining different harvesting strategies and also size estimation methods) and efficient change detection approaches. For each of these challenges, we showed how to improve the state-of-the-art and also provided thorough insights over other methods for further improvements.

Generally speaking, in this work, we proposed focused web harvesting and monitoring approaches based on different query generation mechanisms. In these approaches, to form the best next query to submit, we applied factors like frequency and presence of terms in external corpora and retrieved documents. These factors are important in search engine ranking algorithms. To find other potentially influential factors to include in our focused web harvesting and monitoring approaches, we need to further analyze search engine ranking algorithms. For example, the effects of terms distribution in returned documents, their distances and dependency of candidates to all previously submitted queries need to be studied in counteracting biases generated by search engines.

In addition to the elements of search engine ranking algorithms, the further analysis of the retrieved results by applying techniques such as entity recognition, deduplication and disambiguation and text parsing may help to refine returned results and provide more relevant information to select terms for query reformulation. Future work should also focus on experimenting with the effects of these factors.

7.3 CONCLUDING REMARKS

To move towards achieving a complete coverage in focused web harvesting, especially from websites imposing access limitations, we need to address efficiency in our harvesting methods. To reduce the number of submitted queries while moving towards a complete coverage on relevant information for a given topic, query generation mechanisms are effective. A well-defined query formulation method can enhance efficiency and therefore, coverage on accessing relevant information to our topics of interest. Monitoring web data in focused deep web harvesting can also highly benefit from adapting query generation mechanisms to efficiently detect and retrieve changed content. Further improvements in both data coverage and monitoring tasks can help users such as journalists, business analysts, organizations and governments to reach the data they need without requiring extreme software and hardware facilities. With this thesis, we hope to have contributed to the goal of focused web harvesting and monitoring topics of interest over time.

BIBLIOGRAPHY

[1] Serge Abiteboul. Issues in monitoring web data. In Abdelkader Hameurlain, Rosine Cicchetti, and Roland Traunmüller, editors, *Database and Expert Systems Applications*, volume 2453 of *Lecture Notes in Computer Science*, pages 1–8. Springer Berlin Heidelberg, 2002.

[2] Brad Adelberg. Nodose—a tool for semi-automatically extracting structured and semistructured data from text documents. *SIGMOD Rec.*, 27(2):283–294, June 1998.

[3] Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. A study of distinctiveness in web results of two search engines. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 267–273, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[4] The New York Times Company Alex Wright. Exploring a deep web that google can not grasp. `http://www.nytimes.com/2009/02/23/technology/internet/23search.html?pagewanted=all`, accessed 2012.

[5] Manuel Alvarez, Alberto Pan, Juan Raposo, and Angel Vina. Client-side deep web data extraction. In *Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference*, CEC-EAST '04, pages 158–161, Washington, DC, USA, 2004. IEEE Computer Society.

[6] 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.

[7] Steven Amstrup, Trent McDonald, and Bryan F. Manly. *Handbook of Capture-Recapture Analysis*. Princeton University Press, Princeton, NJ, October 2005.

[8] Aris Anagnostopoulos, Andrei Z. Broder, and David Carmel. Sampling search-engine results. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 245–256, New York, NY, USA, 2005. ACM Press.

[9] Fajar Ardian and Sourav S. Bhowmick. Efficient maintenance of common keys in archives of continuous query results from deep websites. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 637–648, Washington, DC, USA, 2011. IEEE Computer Society.

[10] Ricardo A. Baeza-Yates, Carlos Castillo, and Felipe Saint-Jean. Web dynamics, structure, and page quality. In Mark Levene and Alexandra Poulovassilis, editors, *Web Dynamics - Adapting to Change in Content, Size, Topology and Use*, pages 93–112. Springer, 2004.

[11] Ziv Bar-Yossef and Maxim Gurevich. Efficient search engine measurements. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 401–410. ACM, 2007.

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

[13] Ziv Bar-Yossef and Maxim Gurevich. Efficient search engine measurements. *TWEB*, 5(4):18, 2011.

[14] Luciano Barbosa and Juliana Freire. Siphoning hidden-web data through keyword-based interfaces. In Sérgio Lifschitz, editor, *XIX Simpósio Brasileiro de Bancos de Dados, 18-20 de Outubro, 2004,Brasília, Distrito Federal, Brasil, Anais/Proceedings*, pages 309–321. UnB, 2004.

[15] Pablo Barrio, Luis Gravano, and Chris Develder. Ranking deep web text collections for scalable information extraction. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 153–162, New York, NY, USA, 2015. ACM.

[16] Michael Benedikt, Georg Gottlob, and Pierre Senellart. Determining relevance of accesses at runtime. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '11, pages 211–222, New York, NY, USA, 2011. ACM.

[17] Michael K Bergman. White paper: the deep web: surfacing hidden value. *Journal of electronic publishing*, 7(1), 2001.

[18] V Bharanipriya and V Kamakshi Prasad. Web content mining tools: a comparative study. *International Journal of Information Technology and Knowledge Management*, 4(1):211–215, 2011.

[19] Krishna Bharat and Andrei Z. Broder. A technique for measuring the relative size and overlap of public web search engines. *Computer Networks*, 30(1-7):379–388, 1998.

[20] Robert Boncella. Competitive intelligence and the web. In *9th Americas Conference on Information Systems, AMCIS 2003, Tampa, FL, USA, August 4-6, 2003*, page 418. Association for Information Systems, 2003.

[21] 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, 2000. North-Holland Publishing Co.

[22] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.

[23] Andrei Broder, Marcus Fontura, Vanja Josifovski, Ravi Kumar, Rajeev Motwani, Shubha Nabar, Rina Panigrahy, Andrew Tomkins, and Ying Xu. Estimating corpus size via queries. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 594–603, New York, NY, USA, 2006. ACM.

[24] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 327–336. ACM, 1998.

[25] Michael J. Cafarella. Extracting and querying a comprehensive web database. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*. www.cidrdb.org, 2009.

[26] Andrea Calì and Davide Martinenghi. Querying data under access limitations. In Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 50–59. IEEE Computer Society, 2008.

[27] Andrea Calì and Davide Martinenghi. Querying the deep web. In Ioana Manolescu, Stefano Spaccapietra, Jens Teubner, Masaru Kitsuregawa, Alain Léger, Felix Naumann, Anastasia Ailamaki, and Fatma Özcan, editors, *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, volume 426 of *ACM International Conference Proceeding Series*, pages 724–727. ACM, 2010.

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

[29] 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.

[30] 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.

[31] Carlos Castillo. Effective web crawling. In *ACM SIGIR Forum*, volume 39, pages 55–56. ACM, 2005.

[32] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623–1640, 1999.

[33] Kevin Chen-Chuan Chang, Bin He, Chengkai Li, Mitesh Patel, and Zhen Zhang. Structured databases on the web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.

[34] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR*, pages 44–55, 2005.

[35] 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.

[36] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, April 2002.

[37] 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.

[38] BrightPlanet Corporation. Deep web intelligence. `http://www.brightplanet.com`, accessed 2012.

[39] Valter Crescenzi and Giansalvatore Mecca. Automatic information extraction from large websites. *J. ACM*, 51(5):731–779, September 2004.

[40] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 109–118, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[41] Arjun Dasgupta, Xin Jin, Bradley Jewell, Nan Zhang, and Gautam Das. Unbiased estimation of size and other aggregates over hidden web databases. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 855–866, New York, NY, USA, 2010. ACM.

[42] Victor de Graaff. *Geosocial Recommender Systems*. PhD thesis, Univ. of Twente, Enschede, The Netherlands, December 2015.

[43] Deep web. `http://en.wikipedia.org/wiki/Deep_Web`, accessed 2012.

[44] 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.

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

[46] AnHai Doan, Pedro M. Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In Sharad Mehrotra and Timos K. Sellis, editors, *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, pages 509–520. ACM, 2001.

[47] AnHai Doan, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Managing information extraction: State of the art and research directions. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 799–800, New York, NY, USA, 2006. ACM.

[48] Suelette Dreyfus, Reeva Lederman, Rachelle Bosua, and Simon Milto. Can we handle the truth? whistleblowing to the media in the digital era. *Global Media Journal: Australian Edition*, 5, 2011.

[49] Dynamic web page. `http://en.wikipedia.org/wiki/Dynamic_web_page`, accessed 2013.

[50] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowl. Eng.*, 31(3):227–251, November 1999.

[51] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[52] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *Knowl.-Based Syst.*, 70:301–323, 2014.

[53] What is freeformat. `http://www.gooseeker.com/en/node/knowledgebase/freeformat`, accessed 2013.

[54] The Apache Software Foundation. *The Apache Mahout™*, accessed Aug, 2015.

[55] Dayne Freitag. Machine learning for information extraction in informal domains. *Mach. Learn.*, 39(2-3):169–202, May 2000.

[56] Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, and Christian Schallhart. The ontological key: automatically understanding and integrating forms to access the deep web. *VLDB J.*, 22(5):615–640, 2013.

[57] Wolfgang Gatterbauer. Web harvesting. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 3472–3473. Springer US, 2009.

[58] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 71–80, New York, NY, USA, 2007. ACM.

[59] D. Geiger, D. Heckerman, and C. Meek. Introduction to monte carlo methods. In M. Jordan, editor, *Learning in graphical models*, pages 175–289. Kluwer, 1998.

[60] Google.com. Google custom search. `https://developers.google.com/custom-search/`, accessed 2015.

[61] Google.com. How search works - crawling and indexing. `https://www.google.com/insidesearch/howsearchworks/crawling-indexing.html`, accessed 2015.

[62] James Grimmelmann. The google dilemma. *NYL Sch. L. Rev.*, 53:939, 2008.

[63] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 902–903, New York, NY, USA, 2005. ACM.

[64] Sonali Gupta and Komal Kumar Bhatia. A comparative study of hidden web crawlers. *CoRR*, abs/1407.5732, 2014.

[65] Alon Y. Halevy and Susan McGregor. Data management for journalism. *IEEE Data Eng. Bull.*, 35:7–15, 2012.

[66] Taher H. Haveliwala, Aristides Gionis, Dan Klein, and Piotr Indyk. Evaluating strategies for similarity search on the web. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 432–442, New York, NY, USA, 2002. ACM.

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

[68] Bin He and Kevin Chen-Chuan Chang. Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 217–228, New York, NY, USA, 2003. ACM.

[69] Bin He and Kevin Chen-Chuan Chang. Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Trans. Database Syst.*, 31(1):346–395, March 2006.

[70] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, May 2007.

[71] 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.

[72] Abdelhakim Herrouz, Chabane Khentout, and Mahieddine Djoudi. Overview of web content mining tools. *CoRR*, abs/1307.1024, 2013.

[73] Timothy C Hoad and Justin Zobel. Methods for identifying versioned and plagiarized documents. *Journal of the American society for information science and technology*, 54(3):203–215, 2003.

[74] Jer Lang Hong. Deep web data extraction. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 3420–3427, 2010.

[75] Import.io. `https://import.io/`, accessed 2015.

[76] Faustina Johnson and Santosh Kumar Gupta. Article: Web content mining techniques: A survey. *International Journal of Computer Applications*, 47(11):44–50, June 2012. Full text available.

[77] John C Kern. An introduction to regression analysis. *American Statistician*, 61(1):101–101, 2007.

[78] Mohamamdreza Khelghati, Djoerd Hiemstra, and Maurice Van Keulen. Deep web entity monitoring. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 377–382, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[79] Mohammadreza Khelghati. HarvestED (Harvest Entity related Documents). `http://mohkhe.github.io/HarvestED/`, 2016.

[80] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice Keulen. How much data resides in a web collection: how to estimate size of a web collection. In *Dutch-Belgian Workshop on Information Retrieval (DIR-2013)*, volume 986, pages 42–43. CEUR-WS.org, 2013.

[81] 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.

[82] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Designing a general deep web harvester by harvestability factor. In *Surfacing the Deep and the Social Web*, volume 1310. http://ceur-ws.org/, 2014.

[83] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Harvesting all matching information to a given query from a deep website, 2015.

[84] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Towards complete coverage in focused web harvesting. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '15, pages 65:1–65:9, New York, NY, USA, 2015. ACM.

[85] Mohammadreza Khelghati, Maurice van Keulen, and Djoerd Hiemstra. Designing a general deep web access approach based on a newly introduced factor; harvestability factor (hf). Technical Report TR-CTIT-14-08, Centre for Telematics and Information Technology, University of Twente, Enschede, June 2014.

[86] Mohammadreza Khelghati, Maurice van Keulen, and Djoerd Hiemstra. Efficient web harvesting strategies for monitoring deep web content. Technical Report TR-CTIT-16-05, Centre for Telematics and Information Technology, University of Twente, Enschede, May 2016.

[87] Kiminolabs. `https://www.kimonolabs.com/`, accessed 2014.

[88] Aleksander Kolcz, Abdur Chowdhury, and Joshua Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 605–610, New York, NY, USA, 2004. ACM.

[89] Lefteris Kozanidis. An ontology-based focused crawler. In *Proceedings of the 13th International Conference on Natural Language and Information Systems: Applications of Natural Language to Information Systems*, NLDB '08, pages 376–379, Berlin, Heidelberg, 2008. Springer-Verlag.

[90] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artif. Intell.*, 118(1-2):15–68, April 2000.

[91] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran Soares da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, 2002.

[92] Alberto H.F. Laender, Berthier Ribeiro-Neto, and Altigran S. da Silva. Debye – data extraction by example. *Data and Knowledge Engineering*, 40(2):121 – 154, 2002.

[93] Amy N. Langville and Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ, USA, 2006.

[94] Steve Lawrence and C Lee Giles. Accessibility of information on the web. *Nature*, 400(6740):107–107, 1999.

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

[96] Chen Li and Edward Y. Chang. Answering queries with useful bindings. *ACM Trans. Database Syst.*, 26(3):313–343, 2001.

[97] 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.

[98] Bing Liu and Kevin Chen-Chuan Chang. Editorial: special issue on web content mining. *SIGKDD Explorations*, 6(2):1–4, 2004.

[99] King-Lup Liu, Clement T. Yu, Weiyi Meng, and Adrian Santoso. Discovering the representative of a search engine. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001*, pages 577–579. ACM, 2001.

[100] Wei Liu, Xiaofeng Meng, and Weiyi Meng. Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22(3):447–460, 2010.

[101] Jianguo Lu. Ranking bias in deep web size estimation using capture recapture method. *Data Knowl. Eng.*, 69(8):866–879, 2010.

[102] Jianguo Lu and Dingding Li. Estimating deep web data source size by capture—recapture method. *Inf. Retr.*, 13(1):70–95, February 2010.

[103] Jayant Madhavan, Loredana Afanasiev, Lyublena Antova, and Alon Y. Halevy. Harnessing the deep web: Present and future. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*. www.cidrdb.org, 2009.

[104] Jayant Madhavan, Shawn R. Jeffery, Shirley Cohen, Xin (luna Dong, David Ko, Cong Yu, Alon Halevy, and Google Inc. Web-scale data integration: You can only afford to pay as you go. In *In Proc. of CIDR-07*, 2007.

[105] 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.

[106] Swati Mali and B.B. Meshram. Focused web crawler with page change detection policy. *IJCA Proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET)*, (9):51–56, 2011. Full text available.

[107] 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.

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

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

[110] Robert Meusel, Peter Mika, and Roi Blanco. Focused crawling for structured data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1039–1048, New York, NY, USA, 2014. ACM.

[111] SY Mudhasir, J Deepika, S Sendhilkumar, and GS Mahalakshmi. Near-duplicates detection and elimination based on web provenance for effective web search. *International Journal on Internet and Distributed Computing Systems*, 1(1):22–32, 2011.

[112] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1-2):93–114, March 2001.

[113] Marc Najork. Politeness policies and web crawler architecture. In LING LIU and M.TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 3462–3465. Springer US, 2009.

[114] 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.

[115] Umara Noor, Zahid Rashid, and Azhar Rauf. Article: A survey of automatic deep web classification techniques. *International Journal of Computer Applications*, 19(6):43–50, April 2011. Published by Foundation of Computer Science.

[116] Alexandros Ntoulas, Petros Zerfos, and Junghoo Cho. Downloading textual hidden web content through keyword queries. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '05, pages 100–109, New York, NY, USA, 2005. ACM.

[117] Carlos R. Osuna, Rafael Z. Frantz, David Ruiz-Cortes, and Rafael Corchuelo. On using high-level structured queries for integrating deep-web information sources. In *International Conference on Software Engineering Research and Practice*, pages 630–636, 2011.

[118] Outwit hub. `http://www.outwit.com/`, accessed 2014.

[119] Politeness policy of web crawlers. `https://en.wikipedia.org/wiki/Web_crawler#Politeness_policy`, accessed 2016.

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

[121] Pubmed - us national library of medicine, national institutes of health. `http://www.ncbi.nlm.nih.gov/pubmed`, accessed 2013.

[122] W. Pugh and M.H. Henzinger. Detecting duplicate and near-duplicate files, December 2 2003. US Patent 6,658,423.

[123] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 129–138, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[124] 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.

[125] Berthier Ribeiro-Neto, Alberto H. F. Laender, and Altigran S. da Silva. Extracting semi-structured data through examples. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, pages 94–101, New York, NY, USA, 1999. ACM.

[126] Cheng Sheng, Nan Zhang, Yufei Tao, and Xin Jin. Optimal algorithms for crawling a hidden database in the web. *Proc. VLDB Endow.*, 5(11):1112–1123, July 2012.

[127] 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.

[128] 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.

[129] Aliaksandr Talaika, Joanna Biega, Antoine Amarilli, and Fabian M Suchanek. Ibex: Harvesting entities from the web using unique identifiers. In *Proceedings of the 18th International Workshop on Web and Databases*, pages 13–19. ACM, 2015.

[130] Top 100 websites for your career. `http://www.forbes.com/sites/jacquelynsmith/2013/09/18/the-top-100-websites-for-your-career/`, accessed 2013.

[131] Paul Thomas. Generalising multiple capture-recapture to non-uniform sample sizes. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 839–840, New York, NY, USA, 2008. ACM.

[132] Kien Tjin-Kam-Jet, Dolf Trieschnigg, and Djoerd Hiemstra. Free-text search versus complex web forms. In Paul D. Clough, Colum Foley, Cathal Gurrin, Gareth J. F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Murdock, editors, *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland, April 18-21, 2011. Proceedings*, volume 6611 of *Lecture Notes in Computer Science*, pages 670–674. Springer, 2011.

[133] Antal van den Bosch, Toine Bogers, and Maurice de Kunder. A longitudinal analysis of search engine index size. In Albert Ali Salah, Yasar Tonta, Alkim Almila Akdag Salah, Cassidy R. Sugimoto, and Umut Al, editors, *Proceedings of ISSI 2015 Istanbul: 15th International Society of Scientometrics and Informetrics Conference, Istanbul, Turkey, 29 June to 3 July, 2015*. Bogaziçi University Printhouse, 2015.

[134] Antal van den Bosch, Toine Bogers, and Maurice de Kunder. Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856, 2016.

[135] Web content extractor. `http://www.webcontentextractor.com/`, accessed 2014.

[136] Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In Jan Paredaens and Dirk Van Gucht, editors, *PODS*, pages 65–76. ACM, 2010.

[137] 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.

[138] Daily estimated size of the world wide web. `http://worldwidewebsize.com/`, accessed 2015.

[139] Ping Wu, Ji-Rong Wen, Huan Liu, and Wei-Ying Ma. Query selection techniques for efficient crawling of structured web sources. In Ling Liu, Andreas Reuter, Kyu-Young Whang, and Jianjun Zhang, editors, *ICDE*, page 47. IEEE Computer Society, 2006.

[140] Jingfang Xu, Sheng Wu, and Xing Li. Estimating collection size with logistic regression. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 789–790, New York, NY, USA, 2007. ACM.

[141] Yahoo dapper! `http://open.dapper.net/`, accessed 2014.

[142] Feiyue Ye and Hang Yu. Research on automate discovery of deep web interfaces. In Jianyong Wang, Wojciech Cellary, Dingding Wang, Hua Wang, Shu-Ching Chen, Tao Li, and Yanchun Zhang, editors, *Web Information Systems Engineering - WISE 2015 - 16th International Conference, Miami, FL, USA, November 1-3, 2015, Proceedings, Part II*, volume 9419 of *Lecture Notes in Computer Science*, pages 191–198. Springer, 2015.

[143] Alessandro Zanasi. Competitive intelligence through data mining public sources. *Competitive Intelligence Review*, 9(1):44–54, 1998.

[144] Nan Zhang and Gautam Das. Exploration of deep web repositories. *PVLDB*, 4(12):1506–1507, 2011.

SIKS Dissertation Series

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*
2 Willem Robert van Hage (VUA) *Evaluating Ontology-Alignment Techniques*
3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*
4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks: Based on Knowledge, Cognition, and Quality*
6 Muhammad Subianto (UU) *Understanding Classification*
7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*
8 Volker Nannen (VUA) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*
10 Jan Wielemaker (UvA) *Logic programming for knowledge-intensive interactive applications*
11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*
12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin) *Operating Guidelines for Services*
13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*
14 Maksym Korotkiy (VUA) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
15 Rinke Hoekstra (UvA) *Ontology Representation: Design Patterns and Ontologies that Make Sense*
16 Fritz Reul (UvT) *New Architectures in Computer Chess*
17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*
18 Fabian Groffen (CWI) *Armada, An Evolving Database System*
19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*
20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*
21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*
22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*
23 Peter Hofgesang (VUA) *Modelling Web Usage in a Changing Environment*
24 Annerieke Heuvelink (VUA) *Cognitive Models for Training Simulations*
25 Alex van Ballegooij (CWI) *RAM: Array Database Management through Relational Mapping*
26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*
27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*
28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*

29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*
30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*
31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations from Text*
32 Rik Farenhorst (VUA) *Architectural Knowledge Management: Supporting Architects and Auditors*
33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*
34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*
35 Wouter Koelewijn (UL) *Privacy en Politiegegevens: Over geautomatiseerde normatieve informatie-uitwisseling*
36 Marco Kalz (OUN) *Placement Support for Learners in Learning Networks*
37 Hendrik Drachsler (OUN) *Navigation Support for Learners in Informal Learning Networks*
38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*
39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin) *Service Substitution: A Behavioral Approach Based on Petri Nets*
40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*
41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*
42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*
43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*
44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*
45 Jilles Vreeken (UU) *Making Pattern Mining Useful*
46 Loredana Afanasiev (UvA) *Querying XML: Benchmarks and Recursion*

2010

1 Matthijs van Leeuwen (UU) *Patterns that Matter*
2 Ingo Wassink (UT) *Work flows in Life Science*
3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia documents*
4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*
5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*
6 Sander Bakkes (UvT) *Rapid Adaptation of Video Game AI*
7 Wim Fikkert (UT) *Gesture interaction at a Distance*
8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*
9 Hugo Kielman (UL) *A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging*
10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*
11 Adriaan Ter Mors (TUD) *The world according to MARP: Multi-Agent Route Planning*
12 Susan van den Braak (UU) *Sensemaking software for crime analysis*
13 Gianluigi Folino (RUN) *High Performance Data Mining using Bio-inspired techniques*
14 Sander van Splunter (VUA) *Automated Web Service Reconfiguration*
15 Lianne Bodenstaff (UT) *Managing Dependency Relations in Inter-Organizational Models*
16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, theory and practice*
17 Spyros Kotoulas (VUA) *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*
18 Charlotte Gerritsen (VUA) *Caught in the Act: Investigating Crime by Agent-Based Simulation*
19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*
20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*
21 Harold van Heerde (UT) *Privacy-aware data management by means of data degradation*
22 Michiel Hildebrand (CWI) *End-user Support for Access to Heterogeneous Linked Data*
23 Bas Steunebrink (UU) *The Logical Structure of Emotions*
24 Zulfiqar Ali Memon (VUA) *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*

25 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*
26 Marten Voulon (UL) *Automatisch contracteren*
27 Arne Koopman (UU) *Characteristic Relational Patterns*
28 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*
29 Marieke van Erp (UvT) *Accessing Natural History: Discoveries in data cleaning, structuring, and retrieval*
30 Victor de Boer (UvA) *Ontology Enrichment from Heterogeneous Sources on the Web*
31 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
32 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
33 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*
34 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*
35 Jose Janssen (OU) *Paving the Way for Lifelong Learning: Facilitating competence development through a learning path specification*
36 Niels Lohmann (TUe) *Correctness of services and their composition*
37 Dirk Fahland (TUe) *From Scenarios to components*
38 Ghazanfar Farooq Siddiqui (VUA) *Integrative modeling of emotions in virtual agents*
39 Mark van Assem (VUA) *Converting and Integrating Vocabularies for the Semantic Web*
40 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*
41 Sybren de Kinderen (VUA) *Needs-driven service bundling in a multi-supplier setting: the computational e3-service approach*
42 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
43 Pieter Bellekens (TUe) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
44 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
45 Vincent Pijpers (VUA) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
46 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
47 Jahn-Takeshi Saito (UM) *Solving difficult game positions*
48 Bouke Huurnink (UvA) *Search in Audiovisual Broadcast Archives*
49 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*
50 Peter-Paul van Maanen (VUA) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
51 Edgar Meij (UvA) *Combining Concepts and Language Models for Information Access*

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*
2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*
3 Jan Martijn van der Werf (TUe) *Compositional Design and Verification of Component-Based Information Systems*
4 Hado van Hasselt (UU) *Insights in Reinforcement Learning: Formal analysis and empirical evaluation of temporal-difference*
5 Base van der Raadt (VUA) *Enterprise Architecture Coming of Age: Increasing the Performance of an Emerging Discipline*
6 Yiwen Wang (TUe) *Semantically-Enhanced Recommendations in Cultural Heritage*
7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*
8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*
9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*
10 Bart Bogaert (UvT) *Cloud Content Contention*
11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*
12 Carmen Bratosin (TUe) *Grid Architecture for Distributed Process Mining*
13 Xiaoyu Mao (UvT) *Airport under Control. Multiagent Scheduling for Airport Ground Handling*
14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*

15 Marijn Koolen (UvA) *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*
16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*
17 Jiyin He (UvA) *Exploring Topic Structure: Coherence, Diversity and Relatedness*
18 Mark Ponsen (UM) *Strategic Decision-Making in complex games*
19 Ellen Rusman (OU) *The Mind ' s Eye on Personal Profiles*
20 Qing Gu (VUA) *Guiding service-oriented software engineering: A view-based approach*
21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*
22 Junte Zhang (UvA) *System Evaluation of Archival Description and Access*
23 Wouter Weerkamp (UvA) *Finding People and their Utterances in Social Media*
24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*
25 Syed Waqar ul Qounain Jaffry (VUA) *Analysis and Validation of Models for Trust Dynamics*
26 Matthijs Aart Pontier (VUA) *Virtual Agents for Human Communication: Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*
27 Aniel Bhulai (VUA) *Dynamic website optimization through autonomous management of design patterns*
28 Rianne Kaptein (UvA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*
29 Faisal Kamiran (TUe) *Discrimination-aware Classification*
30 Egon van den Broek (UT) *Affective Signal Processing (ASP): Unraveling the mystery of emotions*
31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*
32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*
33 Tom van der Weide (UU) *Arguing to Motivate Decisions*
34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*
35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*
36 Erik van der Spek (UU) *Experiments in serious game design: a cognitive approach*
37 Adriana Burlutiu (RUN) *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*
38 Nyree Lemmens (UM) *Bee-inspired Distributed Optimization*
39 Joost Westra (UU) *Organizing Adaptation using Agents in Serious Games*
40 Viktor Clerc (VUA) *Architectural Knowledge Management in Global Software Development*
41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*
42 Michal Sindlar (UU) *Explaining Behavior through Mental State Attribution*
43 Henk van der Schuur (UU) *Process Improvement through Software Operation Knowledge*
44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*
45 Herman Stehouwer (UvT) *Statistical Language Models for Alternative Sequence Selection*
46 Beibei Hu (TUD) *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*
47 Azizi Bin Ab Aziz (VUA) *Exploring Computational Models for Intelligent Support of Persons with Depression*
48 Mark Ter Maat (UT) *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*
49 Andreea Niculescu (UT) *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

2012

1 Terry Kakeeto (UvT) *Relationship Marketing for SMEs in Uganda*
2 Muhammad Umair (VUA) *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*
3 Adam Vanya (VUA) *Supporting Architecture Evolution by Mining Software Repositories*
4 Jurriaan Souer (UU) *Development of Content Management System-based Web Applications*
5 Marijn Plomp (UU) *Maturing Interorganisational Information Systems*
6 Wolfgang Reinhardt (OU) *Awareness Support for Knowledge Workers in Research Networks*
7 Rianne van Lambalgen (VUA) *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*

8 Gerben de Vries (UvA) *Kernel Methods for Vessel Trajectories*
9 Ricardo Neisse (UT) *Trust and Privacy Management Support for Context-Aware Service Platforms*
10 David Smits (TUe) *Towards a Generic Distributed Adaptive Hypermedia Environment*
11 J. C. B. Rantham Prabhakara (TUe) *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*
12 Kees van der Sluijs (TUe) *Model Driven Design and Data Integration in Semantic Web Information Systems*
13 Suleman Shahid (UvT) *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions*
14 Evgeny Knutov (TUe) *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*
15 Natalie van der Wal (VUA) *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes*
16 Fiemke Both (VUA) *Helping people by understanding them: Ambient Agents supporting task execution and depression treatment*
17 Amal Elgammal (UvT) *Towards a Comprehensive Framework for Business Process Compliance*
18 Eltjo Poort (VUA) *Improving Solution Architecting Practices*
19 Helen Schonenberg (TUe) *What's Next? Operational Support for Business Process Execution*
20 Ali Bahramisharif (RUN) *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*
21 Roberto Cornacchia (TUD) *Querying Sparse Matrices for Information Retrieval*
22 Thijs Vis (UvT) *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*
23 Christian Muehl (UT) *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*
24 Laurens van der Werff (UT) *Evaluation of Noisy Transcripts for Spoken Document Retrieval*
25 Silja Eckartz (UT) *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*
26 Emile de Maat (UvA) *Making Sense of Legal Text*
27 Hayrettin Gurkok (UT) *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*
28 Nancy Pascall (UvT) *Engendering Technology Empowering Women*
29 Almer Tigelaar (UT) *Peer-to-Peer Information Retrieval*
30 Alina Pommeranz (TUD) *Designing Human-Centered Systems for Reflective Decision Making*
31 Emily Bagarukayo (RUN) *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*
32 Wietske Visser (TUD) *Qualitative multi-criteria preference representation and reasoning*
33 Rory Sie (OUN) *Coalitions in Cooperation Networks (COCOON)*
34 Pavol Jancura (RUN) *Evolutionary analysis in PPI networks and applications*
35 Evert Haasdijk (VUA) *Never Too Old To Learn: On-line Evolution of Controllers in Swarm- and Modular Robotics*
36 Denis Ssebugwawo (RUN) *Analysis and Evaluation of Collaborative Modeling Processes*
37 Agnes Nakakawa (RUN) *A Collaboration Process for Enterprise Architecture Creation*
38 Selmar Smit (VUA) *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*
39 Hassan Fatemi (UT) *Risk-aware design of value and coordination networks*
40 Agus Gunawan (UvT) *Information Access for SMEs in Indonesia*
41 Sebastian Kelle (OU) *Game Design Patterns for Learning*
42 Dominique Verpoorten (OU) *Reflection Amplifiers in self-regulated Learning*
43 Anna Tordai (VUA) *On Combining Alignment Techniques*
44 Benedikt Kratz (UvT) *A Model and Language for Business-aware Transactions*
45 Simon Carter (UvA) *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*
46 Manos Tsagkias (UvA) *Mining Social Media: Tracking Content and Predicting Behavior*
47 Jorn Bakker (TUe) *Handling Abrupt Changes in Evolving Time-series Data*

48 Michael Kaisers (UM) *Learning against Learning: Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*
49 Steven van Kervel (TUD) *Ontologogy driven Enterprise Information Systems Engineering*
50 Jeroen de Jong (TUD) *Heuristics in Dynamic Sceduling: a practical framework with a case study in elevator dispatching*

2013

1 Viorel Milea (EUR) *News Analytics for Financial Decision Support*
2 Erietta Liarou (CWI) *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*
3 Szymon Klarman (VUA) *Reasoning with Contexts in Description Logics*
4 Chetan Yadati (TUD) *Coordinating autonomous planning and scheduling*
5 Dulce Pumareja (UT) *Groupware Requirements Evolutions Patterns*
6 Romulo Goncalves (CWI) *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*
7 Giel van Lankveld (UvT) *Quantifying Individual Player Differences*
8 Robbert-Jan Merk (VUA) *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*
9 Fabio Gori (RUN) *Metagenomic Data Analysis: Computational Methods and Applications*
10 Jeewanie Jayasinghe Arachchige (UvT) *A Unified Modeling Framework for Service Design*
11 Evangelos Pournaras (TUD) *Multi-level Reconfigurable Self-organization in Overlay Services*
12 Marian Razavian (VUA) *Knowledge-driven Migration to Services*
13 Mohammad Safiri (UT) *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*
14 Jafar Tanha (UvA) *Ensemble Approaches to Semi-Supervised Learning Learning*
15 Daniel Hennes (UM) *Multiagent Learning: Dynamic Games and Applications*
16 Eric Kok (UU) *Exploring the practical benefits of argumentation in multi-agent deliberation*
17 Koen Kok (VUA) *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*
18 Jeroen Janssens (UvT) *Outlier Selection and One-Class Classification*
19 Renze Steenhuizen (TUD) *Coordinated Multi-Agent Planning and Scheduling*
20 Katja Hofmann (UvA) *Fast and Reliable Online Learning to Rank for Information Retrieval*
21 Sander Wubben (UvT) *Text-to-text generation by monolingual machine translation*
22 Tom Claassen (RUN) *Causal Discovery and Logic*
23 Patricio de Alencar Silva (UvT) *Value Activity Monitoring*
24 Haitham Bou Ammar (UM) *Automated Transfer in Reinforcement Learning*
25 Agnieszka Anna Latoszek-Berendsen (UM) *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*
26 Alireza Zarghami (UT) *Architectural Support for Dynamic Homecare Service Provisioning*
27 Mohammad Huq (UT) *Inference-based Framework Managing Data Provenance*
28 Frans van der Sluis (UT) *When Complexity becomes Interesting: An Inquiry into the Information eXperience*
29 Iwan de Kok (UT) *Listening Heads*
30 Joyce Nakatumba (TUe) *Resource-Aware Business Process Management: Analysis and Support*
31 Dinh Khoa Nguyen (UvT) *Blueprint Model and Language for Engineering Cloud Applications*
32 Kamakshi Rajagopal (OUN) *Networking For Learning: The role of Networking in a Lifelong Learner's Professional Development*
33 Qi Gao (TUD) *User Modeling and Personalization in the Microblogging Sphere*
34 Kien Tjin-Kam-Jet (UT) *Distributed Deep Web Search*
35 Abdallah El Ali (UvA) *Minimal Mobile Human Computer Interaction*
36 Than Lam Hoang (TUe) *Pattern Mining in Data Streams*
37 Dirk Börner (OUN) *Ambient Learning Displays*
38 Eelco den Heijer (VUA) *Autonomous Evolutionary Art*
39 Joop de Jong (TUD) *A Method for Enterprise Ontology based Design of Enterprise Information Systems*
40 Pim Nijssen (UM) *Monte-Carlo Tree Search for Multi-Player Games*

41 Jochem Liem (UvA) *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*
42 Léon Planken (TUD) *Algorithms for Simple Temporal Reasoning*
43 Marc Bron (UvA) *Exploration and Contextualization through Interaction and Concepts*

2014

1 Nicola Barile (UU) *Studies in Learning Monotone Models from Data*
2 Fiona Tuliyano (RUN) *Combining System Dynamics with a Domain Modeling Method*
3 Sergio Raul Duarte Torres (UT) *Information Retrieval for Children: Search Behavior and Solutions*
4 Hanna Jochmann-Mannak (UT) *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*
5 Jurriaan van Reijsen (UU) *Knowledge Perspectives on Advancing Dynamic Capability*
6 Damian Tamburri (VUA) *Supporting Networked Software Development*
7 Arya Adriansyah (TUe) *Aligning Observed and Modeled Behavior*
8 Samur Araujo (TUD) *Data Integration over Distributed and Heterogeneous Data Endpoints*
9 Philip Jackson (UvT) *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*
10 Ivan Salvador Razo Zapata (VUA) *Service Value Networks*
11 Janneke van der Zwaan (TUD) *An Empathic Virtual Buddy for Social Support*
12 Willem van Willigen (VUA) *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*
13 Arlette van Wissen (VUA) *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*
14 Yangyang Shi (TUD) *Language Models With Meta-information*
15 Natalya Mogles (VUA) *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*
16 Krystyna Milian (VUA) *Supporting trial recruitment and design by automatically interpreting eligibility criteria*
17 Kathrin Dentler (VUA) *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*
18 Mattijs Ghijsen (UvA) *Methods and Models for the Design and Study of Dynamic Agent Organizations*
19 Vinicius Ramos (TUe) *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*
20 Mena Habib (UT) *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*
21 Kassidy Clark (TUD) *Negotiation and Monitoring in Open Environments*
22 Marieke Peeters (UU) *Personalized Educational Games: Developing agent-supported scenario-based training*
23 Eleftherios Sidirourgos (UvA/CWI) *Space Efficient Indexes for the Big Data Era*
24 Davide Ceolin (VUA) *Trusting Semi-structured Web Data*
25 Martijn Lappenschaar (RUN) *New network models for the analysis of disease interaction*
26 Tim Baarslag (TUD) *What to Bid and When to Stop*
27 Rui Jorge Almeida (EUR) *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*
28 Anna Chmielowiec (VUA) *Decentralized k-Clique Matching*
29 Jaap Kabbedijk (UU) *Variability in Multi-Tenant Enterprise Software*
30 Peter de Cock (UvT) *Anticipating Criminal Behaviour*
31 Leo van Moergestel (UU) *Agent Technology in Agile Multiparallel Manufacturing and Product Support*
32 Naser Ayat (UvA) *On Entity Resolution in Probabilistic Data*
33 Tesfa Tegegne (RUN) *Service Discovery in eHealth*
34 Christina Manteli (VUA) *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems*
35 Joost van Ooijen (UU) *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*
36 Joos Buijs (TUe) *Flexible Evolutionary Algorithms for Mining Structured Process Models*
37 Maral Dadvar (UT) *Experts and Machines United Against Cyberbullying*

38 Danny Plass-Oude Bos (UT) *Making brain-computer interfaces better: improving usability through post-processing*
39 Jasmina Maric (UvT) *Web Communities, Immigration, and Social Capital*
40 Walter Omona (RUN) *A Framework for Knowledge Management Using ICT in Higher Education*
41 Frederic Hogenboom (EUR) *Automated Detection of Financial Events in News Text*
42 Carsten Eijckhof (CWI/TUD) *Contextual Multidimensional Relevance Models*
43 Kevin Vlaanderen (UU) *Supporting Process Improvement using Method Increments*
44 Paulien Meesters (UvT) *Intelligent Blauw: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden*
45 Birgit Schmitz (OUN) *Mobile Games for Learning: A Pattern-Based Approach*
46 Ke Tao (TUD) *Social Web Data Analytics: Relevance, Redundancy, Diversity*
47 Shangsong Liang (UvA) *Fusion and Diversification in Information Retrieval*

2015

1 Niels Netten (UvA) *Machine Learning for Relevance of Information in Crisis Response*
2 Faiza Bukhsh (UvT) *Smart auditing: Innovative Compliance Checking in Customs Controls*
3 Twan van Laarhoven (RUN) *Machine learning for network data*
4 Howard Spoelstra (OUN) *Collaborations in Open Learning Environments*
5 Christoph Bösch (UT) *Cryptographically Enforced Search Pattern Hiding*
6 Farideh Heidari (TUD) *Business Process Quality Computation: Computing Non-Functional Requirements to Improve Business Processes*
7 Maria-Hendrike Peetz (UvA) *Time-Aware Online Reputation Analysis*
8 Jie Jiang (TUD) *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*
9 Randy Klaassen (UT) *HCI Perspectives on Behavior Change Support Systems*
10 Henry Hermans (OUN) *OpenU: design of an integrated system to support lifelong learning*
11 Yongming Luo (TUe) *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*
12 Julie M. Birkholz (VUA) *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*
13 Giuseppe Procaccianti (VUA) *Energy-Efficient Software*
14 Bart van Straalen (UT) *A cognitive approach to modeling bad news conversations*
15 Klaas Andries de Graaf (VUA) *Ontology-based Software Architecture Documentation*
16 Changyun Wei (UT) *Cognitive Coordination for Cooperative Multi-Robot Teamwork*
17 André van Cleeff (UT) *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*
18 Holger Pirk (CWI) *Waste Not, Want Not!: Managing Relational Data in Asymmetric Memories*
19 Bernardo Tabuenca (OUN) *Ubiquitous Technology for Lifelong Learners*
20 Loïs Vanhée (UU) *Using Culture and Values to Support Flexible Coordination*
21 Sibren Fetter (OUN) *Using Peer-Support to Expand and Stabilize Online Learning*
22 Zhemin Zhu (UT) *Co-occurrence Rate Networks*
23 Luit Gazendam (VUA) *Cataloguer Support in Cultural Heritage*
24 Richard Berendsen (UvA) *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*
25 Steven Woudenberg (UU) *Bayesian Tools for Early Disease Detection*
26 Alexander Hogenboom (EUR) *Sentiment Analysis of Text Guided by Semantics and Structure*
27 Sándor Héman (CWI) *Updating compressed column-stores*
28 Janet Bagorogoza (TiU) *Knowledge Management and High Performance: The Uganda Financial Institutions Model for HPO*
29 Hendrik Baier (UM) *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*
30 Kiavash Bahreini (OUN) *Real-time Multimodal Emotion Recognition in E-Learning*
31 Yakup Koç (TUD) *On Robustness of Power Grids*
32 Jerome Gard (UL) *Corporate Venture Management in SMEs*
33 Frederik Schadd (UM) *Ontology Mapping with Auxiliary Resources*
34 Victor de Graaff (UT) *Geosocial Recommender Systems*

35 Junchao Xu (TUD) *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction*

2016

1 Syed Saiden Abbas (RUN) *Recognition of Shapes by Humans and Machines*
2 Michiel Christiaan Meulendijk (UU) *Optimizing medication reviews through decision support: prescribing a better pill to swallow*
3 Maya Sappelli (RUN) *Knowledge Work in Context: User Centered Knowledge Worker Support*
4 Laurens Rietveld (VUA) *Publishing and Consuming Linked Data*
5 Evgeny Sherkhonov (UvA) *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers*
6 Michel Wilson (TUD) *Robust scheduling in an uncertain environment*
7 Jeroen de Man (VUA) *Measuring and modeling negative emotions for virtual training*
8 Matje van de Camp (TiU) *A Link to the Past: Constructing Historical Social Networks from Unstructured Data*
9 Archana Nottamkandath (VUA) *Trusting Crowdsourced Information on Cultural Artefacts*
10 George Karafotias (VUA) *Parameter Control for Evolutionary Algorithms*
11 Anne Schuth (UvA) *Search Engines that Learn from Their Users*
12 Max Knobbout (UU) *Logics for Modelling and Verifying Normative Multi-Agent Systems*
13 Nana Baah Gyan (VU) *The Web, Speech Technologies and Rural Development in West Africa: An ICT4D Approach*
14 Ravi Khadka (UU) *Revisiting Legacy Software System Modernization*
15 Steffen Michels (RUN) *Hybrid Probabilistic Logics: Theoretical Aspects, Algorithms and Experiments*
16 Guangliang Li (UvA) *Socially Intelligent Autonomous Agents that Learn from Human Reward*
17 Berend Weel (VUA) *Towards Embodied Evolution of Robot Organisms*
18 Albert Meroño Peñuela (VUA) *Refining Statistical Data on the Web*
19 Julia Efremova (TUe) *Mining Social Structures from Genealogical Data*
20 Daan Odijk (UvA) *Context & Semantics in News & Web Search*
21 Alejandro Moreno Célleri (UT) *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground*
22 Grace Lewis (VU) *Software Architecture Strategies for Cyber-Foraging Systems*
23 Fei Cai (UvA) *Query Auto Completion in Information Retrieval*
24 Brend Wanders (UT) *Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach*
25 Julia Kiseleva (TUe) *Using Contextual Information to Understand Searching and Browsing Behavior*
26 Dilhan Thilakarathne (VU) *In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains*
27 Wen Li (TUD) *Understanding Geo-spatial Information on Social Media*
28 Mingxin Zhang (TUD) *Large-scale Agent-based Social Simulation - A study on epidemic prediction and control*
29 Nicolas Höning (CWI/TUD) *Peak reduction in decentralised electricity systems - Markets and prices for flexible planning*
30 Ruud Mattheij (UvT) *The Eyes Have It*
31 Mohammad Khelghati (UT) *Deep web content monitoring*

SUMMARY

Data is one of the keys to success. Whether you are a fraud detection officer in a tax office, a data journalist or a business analyst, your primary concern is to access all the relevant data to your topics of interest. In such an information-thirsty environment, accessing every source of information is valuable. This emphasizes the role of the web as one of the biggest and main sources of data. In accessing web data through either general search engines or direct querying of deep web sources, the laborious work of querying, navigating results, downloading, storing and tracking data changes is a burden on shoulders of users. To decrease this intensive labor work of accessing data, (semi-)automatic harvesters have a crucial role. However, they lack a number of functionalities that we discuss and address in this work.

In this thesis, we investigate the path towards a focused web harvesting approach which can automatically and efficiently query websites, navigate through results, download data, store it and track data changes over time. Such an approach can also facilitate users to access a complete collection of relevant data to their topics of interest and monitor it over time. To realize such a harvester, we focus on the following obstacles.

First, we try to find methods that can achieve the best coverage in harvesting data for a topic. Although using a fully automatic general harvester facilitates accessing web data, it is not a complete solution to collect a thorough data coverage on a given topic. Some search engines, in both surface web and deep web, restrict the number of requests from a user or limit the number of returned results presented to him. We suggest an efficient approach which can pass these limitations and achieve a complete data coverage.

Second, we investigate reducing the cost of harvesting a website regarding the number of submitted requests by estimating its actual size. Harvesting tasks continue till they face the posed query submission limitations by search engines or consume all the allocated resources. To prevent this undesirable situation, we need to know the size of the targeted source. For a website that hides the true size of its residing data, we suggest an accurate method to estimate its size.

As the third challenge, we focus on monitoring data changes over time in web data repositories. This information is helpful in providing the most up-to-date answers to information needs of users. The fast evolving web adds extra challenges for having an up-to-date data collection. Considering the costly process of harvesting, it is important to find methods which facilitate efficient re-harvesting processes.

Lastly, we combine our experiences in harvesting with the studies in the literature to suggest a general designing and developing framework for a web harvester. It is important to know how to configure harvesters so that they can be applied to different websites, domains and settings.

These steps bring further improvements to data coverage and monitoring functionalities of web harvesters and can help users such as journalists, business analysts, organizations and governments to reach the data they need without requiring extreme software and hardware facilities. With this thesis, we hope to have contributed to the goal of focused web harvesting and monitoring topics over time.

SAMENVATTING

Data is een van de sleutels tot succes. Ongeacht wat je taak is, bijvoorbeeld fraude-officier van de belastingdienst, journalist of business analist, je hoofddoel is toegang krijgen tot alle relevante data voor het onderwerp waar je mee bezig bent. In een dergelijke informatie-gedreven omgeving is het zeer waardevol om toegang tot elke databron te hebben. Dit benadrukt de rol van het web als een van de grootste en meest gebruikte databronnen. Het bevragen van websites, navigeren van resultaten, downloaden, opslaan en bijhouden van veranderingen is een zware taak bij het inzien en gebruiken van webdata via generieke zoekmachines of directe queries op deep web sources. Semi-automatische harvesters hebben een cruciale rol in het verlagen van dit intensieve, handmatige werk bij het verkrijgen van data. Deze harvesters missen echter een aantal functionaliteiten, welke we bediscussiëren en aanpakken in dit onderzoek.

In dit onderzoek richten we ons op een aanpak van focused web harvesting, waarmee websites automatisch en efficiënt bevraagd kunnen worden, resultaten genavigeerd worden, data gedownload en opgeslagen kan worden en verandering over tijd bijgehouden kunnen worden. Deze aanpak maakt het voor gebruikers mogelijk om toegang te krijgen tot een volledige verzameling van de voor hen relevante data, en om deze in de gaten te houden over langere periodes. Om deze harvester te ondersteunen richting we ons op de volgende obstakels.

Allereerst vinden we methoden die de hoogste dekkingsgraad leveren bij het harvesten van de data van een bepaald onderwerp. Alhoewel een volledige geautomatiseerde generieke harvester een grote vooruitgang is ten opzicht van handmatig werk, is het nog steeds geen oplossing om een hoge dekkingsgraad te verkrijgen. Sommige zoekmachines, in zowel surface als deep web, limiteren het aantal zoekresultaten dat aan een user getoond wordt. We suggereren een efficiënte aanpak waarmee deze beperkingen overkomen kunnen worden om een volledige dekkingsgraad te bereiken.

Ten tweede onderzoeken we het verlagen van de kosten van het harvesten van een website, met betrekking tot het aantal requests dat gedaan moet worden, door te schatten hoe groot de website daadwerkelijk is. Harvesting gaat normaliter door tot er beperkingen opgelegd worden door de zoekmachine of tot alle bruikbare hulpbron-

nen opgebruikt zijn. Om deze onwenselijke situatie te voorkomen is het nodig om de grootte van de gebruikt bron te weten. We stellen een accurate methode voor om grootteschattingen te maken voor websites die hun werkelijke grootte verhullen.

De derde uitdaging is het bijhouden van veranderingen over tijd. Veranderingsdata is zeer waardevol bij het beantwoorden van de meeste up-to-date antwoorden voor de informatiebehoeften van gebruikers. De snelle evolutie van het web voegt een extra uitdaging toe aan het bijhouden van een up-to-date verzameling. In achte genomen dat het harvesten zelf een kostbaar proces is, is het van groot belang dat methoden gevonden worden voor het efficiënt herharvesten.

Afsluitend combineren we onze ervaringen betreffende webharvesting met onderzoek uit de literatuur om een suggestie te doen voor een ontwerp- en ontwikkel framework voor harvesters. Het is belangrijk om te weten hoe harvesters geconfigureerd kunnen worden om toepasbaar te zijn op verschillende websites, kennisdomeinen en omgevingen.

Deze vier stappen dragen bij aan de verdere verbetering de dekkingsgraad en het bijhouden van veranderingen. Hiermee worden gebruikers zoals journalisten, business analists, organisaties en overheden geholpen om de de data die ze nodig hebben te verkrijgen zonder extreme eisen aan software- en hardwarefaciliteiten. Met deze thesis hopen we een contributie te hebben geleverd aan focused web harvesting en het bijhouden van veranderingen.

Acknowledgements

In your whole life, you only get a few moments to show your gratitude to all the important people in your life in a formal setting. When I look back to select the ones who helped me to stand where I am now, I come up with a very long list: from my friends whose friendship dates back to the first day of school to people whom I got to know during my PhD.

First of all, I would like to express my great appreciation to my promoter, Peter, and my supervisors, Djoerd and Maurice, for giving me the chance to be a member of the Databases group. With your leadership, the working days in a great atmosphere were always pleasant and motivating. Dear Djoerd and Maurice, I greatly appreciate your support, your patience and the inspiration that kept me going further. Without your help, I could not have reached the end of this path. Victor, Juan, Ida, Mena, Brend, Suse, Iwe, Jan and Robin, thank you for keeping up the great atmosphere of the group. Thank you Brend for beating Google in translating the summary of my thesis. I would also like to offer my special thanks to Ida and Suse who were always there when I needed a help with the bureaucratic part of this journey.

Besides the Databases group, associations were a great part of my PhD life, contributing to my well-being and happy mood. I would like to thank P-NUT board members for all the fun and experiences we had together. Juan and Victor, thank you for motivating me to join P-NUT and thanks to Nana who made the board meetings always more pleasant. We became close friends and I always enjoyed our talks and catch-up meetings. Dear Silja, Juan Carlos, Harmen, Febriyani, Rense, Anja, Björn, Janne, Burcu, Jonathan, Marian, Sarah, Imke, Wouter and Adithya, I enjoyed working with you in P-NUT.

I would also like to extend my gratitude to all IrNUTers who joined me in creating something great in our small community in Enschede, a symbol of unity despite all our differences. I should extend my highest appreciations to Mojtaba and Hamed who stood by my side from the very first day. Thank you IrNUTers for all the great moments and experiences which immediately come to my mind whenever I think about my good time in Enschede. My special thanks are extended to Sadaf, Niloofar, Davood, Sara,

Navid and Hassan without who, IrNUT wouldn't have existed and progressed.

I also want to thank all my friends, from the Netherlands, Sweden, Norway, Austria, Iran, Australia and USA, for all the "being there for me"s and for all the love they gave me which I appreciate the most in my life. My best friends, my brothers, Ehsan, Farzad, Adel, Behzad, Vahid, Mohammad and Armin, with whom my friendship ages more than 20 years, we grew up together and shared the most. Despite the distance, every time we meet, it seems we just met the day before. Your friendship always brings joy to my life.

I started this journey from Stockholm with Meysam, a great friend and the best housemate I've ever had, Amirhossein, *mardomitariin chehre*, Farbod, whose success stories always made me happy, dearest Shaghayegh, who taught me the most about friendship, Lale, a patient girl whose friendship is special to me, dearest Bita, my close friend Shohre, *golam* Azade, *behtarin aroos* Miganoush, Samsam, Elnaz, Pouria, *javoon avval* of Scandinavia, Amir, who made Vienna even more pleasant with his great companionship, Mina, whose laugh is contagious, my good friend Farzad Abtahi, Kambiz and Farahnaz, the lovely couple with whom the moments are always happier, Armin, Shafagh and Alireza. I cannot imagine how my life would have turned out to be without your friendship. Thank you for making me think of Stockholm as my home.

Sadaf "Suarez" and Mojtaba Farmanbar, it is always fun to see the flame of your love. Thanks for "always having everything", for the great moments and talks around the always great food. Above all, thanks for your friendship. It means the world to me. I am looking forward to welcome you in Leiden and build even more good memories. Siavash and Mitra, the best bed and breakfast hosts, my life in Enschede would not have had half the fun without Sia's calls and Mitra's sleep during cinema nights. Thank you Siavash for always having my back in lastige situaties. Hamed, you are a generous friend who never said no whenever I needed you. I wish you and Maryam, a happy life with your dear Reyhane. Davood-Marco Polo, my friend and my successor in IrNUT, thanks for your help and friendship.

I would also like to thank my friends in Enschede, the best neighbors, Mozhdeh and Morteza and Mohammad and Neda, quiet Nazli, Behnam, the best *kabab-zan* and my teacher in making kabab, Amir, Alex, Adriana, Afshin, Mohammad, Shahin, Alireza, Meisam, Taher, Damon, Saghar, Foad, Helia, Kay and Saeed whose companion made Enschede a much better place for me to live. My special thanks go to Kambiz, Farahnaz, Parisa and Mani with whom I had amazing parties and great barbecues in the Netherlands. Talking about barbecues, I should definitely thank Björn and his crew for taking my brewery and barbecuing expertise to the next level.

Nick and Corina, thank you for making the salsa lessons so much fun, for the good food and drinks together and all the laughs. Portuguese team, Joana, David and Lionel, I thank you for all the good food you fed me with, from the great flamed *chourico* to the tasty cheese, the homemade *folar* and the amazing *ginjinha*. I extend my acknowledgements to my other Portuguese friends Ana, Diana, Joao and Freskinha for their great hospitality and making my trips to Portugal even more special.

Acknowledgements

Victor, I already mentioned your name two or three times in this section. This shows how grateful I am to get to know you in the Netherlands. First, you were my colleague showing me the fun Dutch videos in the office, then you became my predecessor in P-NUT, and now you are my paranymph and a very close friend. Thanks for all your technical, administrative, and personal help and advice. I am waiting for the day that your PS skills can match mine. Maybe, someday I can run a marathon with you, just kidding.

Ana, I am grateful the most to Enschede and this PhD because of you. A beautiful girl from the north of Portugal whose smile removes all the troubles from my mind. I hear often that patience and northern Portuguese women do not come together, but not only you have been so patient with me, you also made me calm in my hard times in this PhD. Your love and your pleasant company have played a major role in my happiness during this PhD. Without you, I could have never reached this final step. Thanks for all your support and love.

Last but definitely not the least, I want to thank my parents, my sister and Arash for all their support and encouragement throughout my studies. Madar, Pedar and my dear Sanaz, you have believed in me in times that I myself did not even think that I can accomplish even a small bit. Your unconditional love and never-ending patience always make me astonished and empower me to go further and further. The distance and the homesickness disappear when I hear your voice. I am so grateful to have you three with me in this journey.

During the last years, his passion in computer science took Mohammadreza to Sweden, Germany, Austria and the Netherlands, each adding to the diversity of his life experiences, education and skills. After his bachelor's studies at IASBS University, Iran, he moved to Sweden to follow his master's studies in Engineering and Management of Information Systems at KTH University. For his master thesis, he moved to I5 Institute at RWTH University, Aachen, focusing on information retrieval domain. During his PhD at Databases group of the University of Twente, he focused on monitoring web data by bringing together information retrieval and information extraction domains.

ISBN:978-90-365-4123-7

Mohammadreza Khelghati