The Journal of Supercomputing (2020) 76:8175–8192
https://doi.org/10.1007/s11227-019-02787-9

An effective approach to enhancing a focused crawler using Google

Jae-Gil Lee[1] · Donghwan Bae[1] · Sansung Kim[1] · Jungeun Kim[1] · Mun Yong Yi[1]

Published online: 20 February 2019

Abstract

In this paper, we share our experience in augmenting a focused crawler of our vertical search engine designed to work with academic slides. The goal of the *focused* crawler was to collect Microsoft PowerPoint files from academic institutions. A previous approach based on a *general* web crawler can fail to collect a sufficient number of files mainly because of the robots exclusion protocol and missing hyperlinks. As a remedy to these problems, we propose a combinatory approach in which the indexing information maintained by a general web search engine such as Google is utilized for target URL list generation through our query generator, further then complemented by our URL extractor and file downloader. Because Google has already crawled billions of web pages, it will be more cost-efficient and potentially effective to systematically retrieve the desired information from Google than to redo crawling from scratch by ourselves. Our focused crawler, which we call *SlideCrawler*, has been used for our vertical search engine *CourseShare* since the fall of 2011. The capability of SlideCrawler was verified for the top-500 world wide universities. SlideCrawler collected about one million files from the top-500 universities. Further, the study results show that SlideCrawler outperforms Nutch, collecting 3.7 times more slide files.

Keywords Web crawler · Focused crawler · Google · Vertical search engine

✉ Jae-Gil Lee
jaegil@kaist.ac.kr

Donghwan Bae
bdhwan@kaist.ac.kr

Sansung Kim
sansung@kaist.ac.kr

Jungeun Kim
je_kim@kaist.ac.kr

Mun Yong Yi
munyi@kaist.ac.kr

[1] Graduate School of Knowledge Service Engineering, KAIST, Daejeon, Republic of Korea

1 Introduction

Entering the era of Big Data, we are experiencing unprecedented growth of data and resources on the Web. It has been predicted that the amount of data being produced annually will be 44 times greater in 2020 than it was in 2009 [12]. Search engines such as Google, Bing, and Yahoo are being widely used to find and access the information users are in need of. These *general* search engines return satisfactory results in most cases, but the precision of the results is known to improve when the scope of a query is limited to specific domains [6].

A *vertical* search engine, as distinct from a general web search engine, focuses on a specific segment of online contents. The vertical content area may be based on the topicality, media type, or genre of contents [28]. For example, as for the topicality, WebMD (http://www.webmd.com/) is a vertical search engine for medical issues; as for the media type, SlideShare (http://www.slideshare.net/) or SlideFinder (http://www.slidefinder.net/) for PowerPoint files; as for the genre, Microsoft Academic Search (http://academic.microsoft.com/) for academic contents. As the types and sources of the data are very diverse these days, it is expected that vertical search engines will become more popular to pinpoint a piece of domain-specific information on the Web.

A *web crawler* is a computer program that browses the Web in an automated manner to collect the contents for a search engine. It plays a vital role inside a search engine for finding new web pages to be indexed as well as for periodically re-crawling and updating the index with fresh information [17]. A *focused crawler* is a special-purpose web crawler that attempts to download only the web pages that are relevant to a pre-defined topic or set of topics [27]. Because vertical search engines typically use a focused crawler, there has been active research on a focused crawler in the research community [4–6, 10, 20, 22, 29].

One of the major problems of a focused crawler is to effectively order the links at the *crawl frontier*, i.e., the outlinks of already visited web pages, in order to maximize the number of relevant pages while minimizing the number of irrelevant pages. This problem is challenging because possibly many irrelevant pages could exist on the way to relevant pages. Thus, various crawling strategies have been reported in the literature. For example, Chakrabarti et al. [4] used a mix of the relevance of the text and the centrality of the page; Diligenti et al. [10] used the context graphs of seed pages.

In this paper, we present our approach to enhancing a focused crawler for our vertical search engine. CourseShare is our on-going effort to develop a vertical search engine for *academic slides*. That is, the scope of CourseShare is confined to a specific media type (slides) and genre (academic contents), but not for a specific topic. Our focused crawler, named as *SlideCrawler*, is being used for the CourseShare project since the fall of 2011.

The goal of our focused crawler is to efficiently collect slide files (i.e., `ppt` and `pptx` files) as much as possible from university web sites. Typically, for this kind of a focused crawler, a general web crawler has been adapted such that it extracts the links of those files from the web pages crawled. Our experience, however, indicates

that this previous approach suffers from the problem of an insufficient number of crawled files mainly because of the robots exclusion protocol [15] and missing hyperlinks. As a remedy to this problem, we propose to use a *general* search engine such as Google to fetch a list of the URLs of the downloadable files on the Web. Since, in our approach, the general web search engine takes care of complicated tasks, the proposed focused crawler has the following benefits.

- Our focused crawler is very easy to implement. The software consists of only six thousand lines of Java codes.
- Our focused crawler can collect more files than a simple variation of a general web crawler. The experiment result showed that ours collected 3.7 times more slides than the general one from the world wide top-20 universities.
- Incremental crawling, which collects only *updated* files, can be supported easily.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents our design criteria and detailed approach to our focused crawler. Section 4 presents the execution results of our focused crawler. Finally, Sect. 5 concludes this study.

2 Related work

2.1 Focused crawlers

Web crawlers have been heavily studied since late 90s because they are one of the essential components of a search engine. Previous studies can be roughly grouped into three popular topics. First, there has been significant research efforts devoted to a *focused crawler* [4–6, 10, 20, 22, 29]. Second, it is important to keep the collection of web pages up-to-date with a smaller number of re-visits. Thus, *incremental crawling*, which fetches only updated pages, has received a lot of attention from the research community [7, 8, 11]. Last, *distributed and parallel crawling* have been adopted in many platforms to expedite crawling procedures [1, 11, 13, 24]. Our study is closely related to the first topic.

As for a focused crawler, most studies have focused on the methods that predict if a web page is relevant to a given topic or set of topics *before* downloading it to a local database. For this reason, a focused crawler is also called a *topical* crawler. Such methods are mainly classified into two categories: content-based and link-based [6]. The *content-based* category applies text mining and keyword extraction techniques to help determine whether a web page's content is within a specific domain. The methods in this category usually check if the words on a web page overlap a list of domain-specific terminology. The second category exploits the link structure on the Web to infer the information of web pages. The *anchor text*, which is the word or phrase that hyperlinks to a target page, is known to be useful for this prediction since it often contains the description of the target page. The methods in this category use either the anchor text or the text near it to predict a target page's

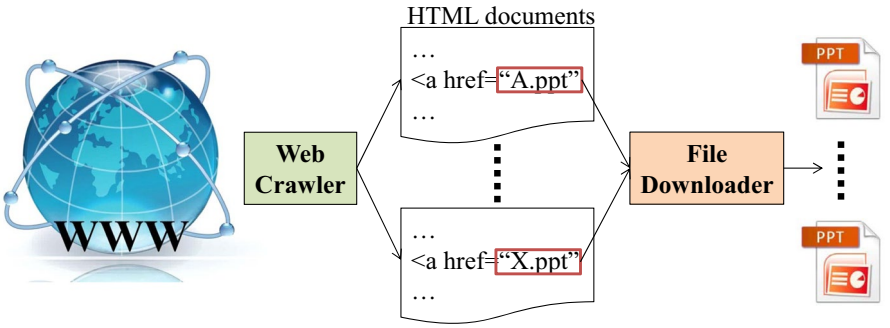


Fig. 1 A typical approach to crawling the files of a specific media type

content. Our study tackles a different aspect of focused crawling in that our crawling is not confined to a specific topic but to a specific media type.

Using a general search engine for focused crawling is not a new idea. One major usage is for obtaining a good set of seed URLs (starting URLs) about a given topic [22, 26]. Then, a focused crawler starts from those seed URLs and collects the relevant web pages by following outlinks. Another major usage is for obtaining a good set of representative terms on a given topic [10, 20, 26, 29]. These terms are then used to predict the relevance of a web page to the topic. Overall, these approaches aim at effectively narrowing down the scope of focused crawling or improving the accuracy of relevance prediction. In our study, we use a general search engine for focused crawling *in a new way* because our focused crawler is confined to a specific media type and not to a specific topic.

The focused crawler for our purpose—crawling the files of a specific media type—might have been developed elsewhere considering that there are several sites similar to CourseShare. Even though we could not find a reference to such a method in academic publications, we conjecture that the most common way is to modify a general web crawler such as Apache Nutch[1] so that it extracts the URLs of those files from the web pages crawled. Then, these URLs are fed to a file downloader. This procedure is illustrated in Fig. 1. We discuss the limitations of this approach in Sect. 3.1.

2.2 Recent trends

Boukadi et al. [3] proposed a focused crawler, called FC4CD, for cloud services. FC4CD and SlideCrawler have commonality in that both of them rely on search engines to get the URLs of target data. However, the purpose of FC4CD is distinct from that of SlideCrawler: the former collects the web pages relevant to cloud providers and services, whereas the latter collects the PowerPoint files from

[1] Apache Nutch is an open source web-search software project, and its project homepage is http://nutch.apache.org/. Its crawler has been written from scratch specifically for this project.

academic institutions. In fact, the main focus of FC4CD is how to populate the proposed ontology from the web pages crawled; thus, the details on how to use search engines are not presented. On the contrary, SlideCrawler is fully reproducible, because the details on how to use Google for query generation and URL extraction are clarified in this paper. Furthermore, SlideCrawler had been developed earlier in the fall of 2011.

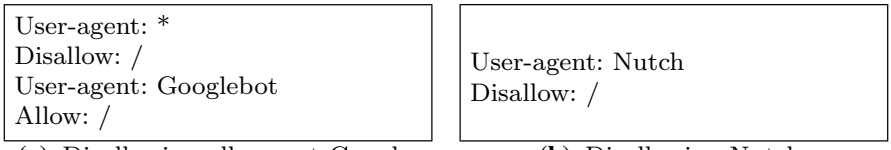
Zhao et al. [30] proposed a focused crawler, called SmartCrawler, for the deep (or hidden) Web. SmartCrawler also uses Google for site locating. Given a seed site link, it retrieves all web pages pointing to the link using Google's `link:` facility. In addition, Diligenti et al. [10] used the same facility to build a context graph of a seed document, which is used for constructing a topic classifier. Thus, their way of using Google is clearly different from ours. Vieira et al. [26] developed a system, called BFC, which constructs and issues queries to a search engine to obtain a diverse and representative set of seeds. Again, the purpose of BFC is distinct from that of SlideCrawler, because the focus of BFC is to expand a set of query keywords relevant to a given topic.

A few studies have taken advantage of graph theory. Bonato et al. [2] proposed a simplified model for crawling complex networks, which is based on the graph cleaning process. Kleinberg et al. [14] investigated more generally the problem of decentralized search in networks, which can be applied to focused crawlers. On the other hand, some researchers have focused on the evaluation of crawlers. Menczer et al. [19] reported the evaluation results of various topical crawlers using two performance metric: recall level and mean similarity. It is hard to directly apply this evaluation framework to our work, because we crawl PowerPoint files other than web pages. Tatli et al. [25] developed a benchmark tool, called WIVET, for analyzing the coverage of the crawler component of a web application vulnerability scanner. Last, there have been research efforts for new data and platforms. Shemshadi et al. [23] developed a crawler for Internet-of-Things (IoT) data on the Web, rather than ordinary web pages. Lee et al. [18] suggested an interesting approach to mobile web search that gathers "inclusive" information about a given query. This approach returns a hierarchical structure of web pages such that users can easily navigate the information on mobile devices.

3 Design criteria and approach

3.1 Limitations of the previous approach

Although the typical approach in Fig. 1 "looks" reasonable, we encountered the problem of an insufficient number of crawled files with this approach. The approach inherits the weakness of general web crawlers. Consequently, the problem is mainly caused by the robots exclusion protocol and missing hyperlinks.



(a) Disallowing all except Google. **(b)** Disallowing Nutch.

Fig. 2 Examples of robots.txt

3.1.1 Robots exclusion protocol

The robots exclusion protocol is a convention that enables server administrators to instruct web crawlers. A text file named "robots.txt" is placed at the root of a web server. Then, a web crawler reads the file and conforms to the instructions in the file. A robots.txt file selectively allows or disallows web crawlers to collect the contents of the directories specified. Figure 2 shows two examples of robots.txt: Fig. 2a for disallowing all web crawlers except Google and Fig. 2b for disallowing Nutch. Please note that the protocol is purely advisory and thus relies on the cooperation of web crawlers.

At the initial stage of our development employing the previous approach, we experienced that almost no web pages were crawled from a specific site even though the site was correctly operating. In many such cases, the robots.txt file allowed only the web crawlers from well-known search engines such as Google. Thus, a custom web crawler or even Nutch could not retrieve web pages from the site. A workaround is to just ignore the robots.txt file, but it is, of course, unethical.

3.1.2 Missing hyperlinks

Because general web crawlers expand their coverage by traversing hyperlinks, the absence of hyperlinks to some web pages implies that they won't be collected by the web crawlers. Figure 3 illustrates two possible reasons for missing hyperlinks. First, if some server (Server **2** in the figure) is temporarily unavailable at the time a web crawler tries to visit there, the web pages beyond the server cannot be retrieved by the web crawler. Second, if some existing hyperlinks (from Server **3** to Server **6** in the figure) have been removed probably unintentionally, the entire part reached through them is not reachable any longer. We actually found that, in several cases, a path did *not* exist between a seed URL and a target page that was expected to be retrieved but was not retrieved.

This problem can be partially solved by providing a larger number of more detailed seed URLs—i.e., by pushing down the starting points deeper on the Web. In this way, we are able to make those seed URLs closer to target pages potentially having the desired files. For example, a university site http://www.mit.edu/ may be replaced with its laboratories such as http://www.media.mit.edu/, http://www.ll.mit.edu/, and http://www.csail.mit.edu/. However, it is not straightforward to find such smaller web sites *comprehensively* from a given Internet domain. In

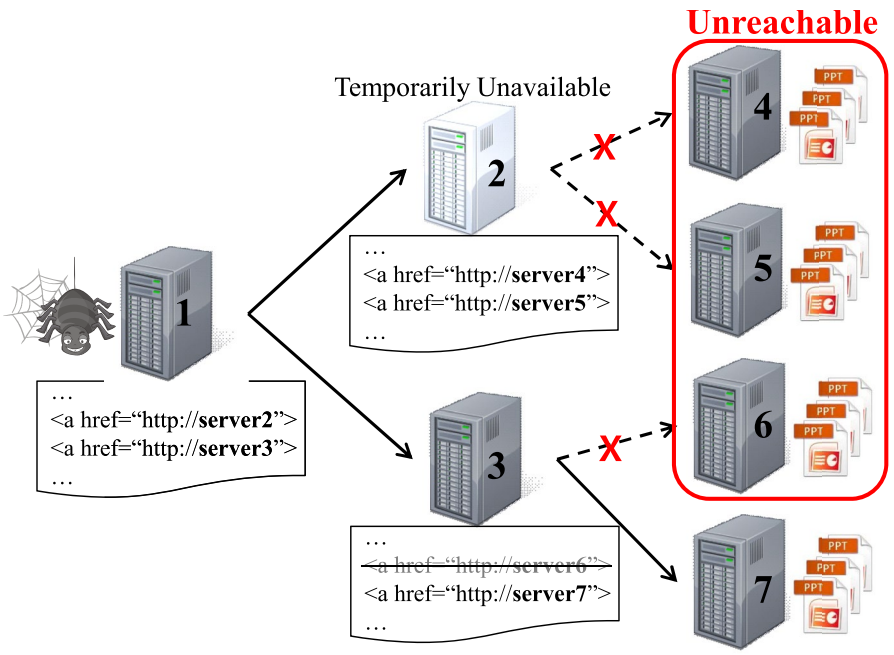


Fig. 3 Two possible reasons for missing hyperlinks

addition, increasing the number of seed URLs will increase the chances of encountering a crawler trap [21], causing a crawler to crash or to make an infinite number of requests.

3.2 Overview of the new approach

Our new approach benefits from a general web search engine such as Google, Bing, or Yahoo. The rationale behind our design is that these commercial search engines have already crawled billions of web pages (with the files on the pages), and thus, we believe that it is more reasonable to systematically retrieve the desired information from them than to *redo* crawling from scratch by ourselves. Although our approach can work with any search engine, Google was selected for our development since it has been known to store the most number of web pages [16].

It should be noted that our approach does not suffer from the limitations of the previous approach in Sect. 3.1. First, because web sites allow well-known crawlers such as Googlebot, Bingbot, and Slurp to collect their information, most web pages should have been indexed by popular general web search engines. Second, once web pages are indexed in a search engine, they are usually kept indexed unless the site owners explicitly request the removal of the pages. Thus, the web pages with no connections can still be retrieved.

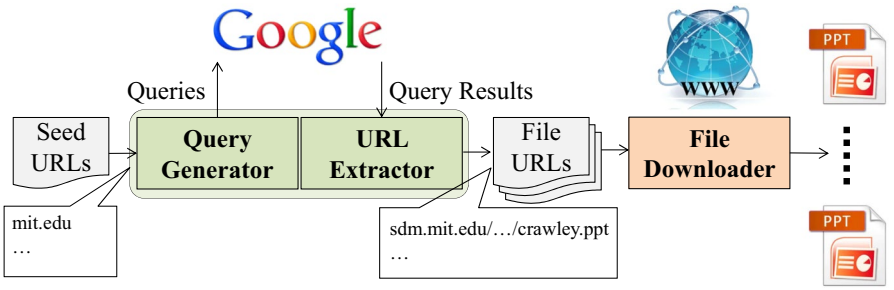


Fig. 4 The overall procedure of our new focused crawling

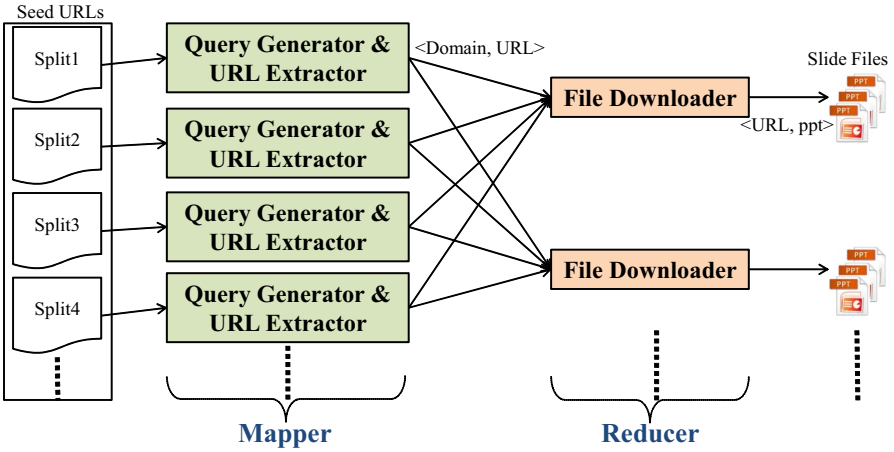


Fig. 5 The parallel architecture of our focused crawler

Figure 4 shows the overall procedure of our new focused crawling. The focused crawler *SlideCrawler* consists of three main components.

1. The *query generator* composes the queries for extracting the URLs of the slide files in a given university and issues these queries to Google. Section 3.3.1 elaborates on this component.
2. The *URL extractor* parses the query results and extracts the URLs of the slide files retrieved. Then, it combines these URLs and passes them to the next component. Section 3.3.2 elaborates on this component.
3. The *file downloader* downloads the files pointed by the URLs and completes the crawling procedure. Section 3.3.3 elaborates on this component.

An important advantage is that our focused crawler can be easily parallelized as in Fig. 5. The query generator and URL extractor run separately from the file downloader. The output of the URL extractor needs to be consumed by the file

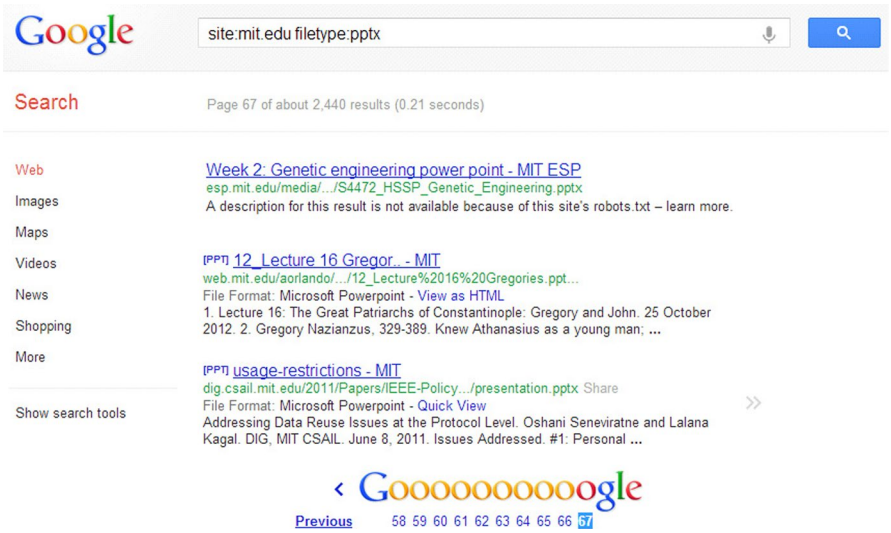


Fig. 6 The last page of Google search results

downloader, and no other communication is necessary. The tasks of query generation and URL extraction run independently of each other if the input (i.e., the seed URLs) is divided into each task. So do the tasks of file downloading. Therefore, our approach is naturally translated into the MapReduce framework [9]. Here, query generation and URL extraction are processed by mappers, and file downloading by reducers. This parallelization significantly improves the performance of crawling when there are many seed URLs or many files to download.

3.3 Implementation of the focused crawler

3.3.1 Query generation

The goal here is threefold, which is to retrieve (1) only the slide files (`ppt` and `pptx`) (2) from a given university (3) exhaustively. Google allows us to specify a file type and limit the search scope to a given Internet domain. The first part of the goal can be achieved by the syntax `filetype:ppt` and `filetype:pptx`. With an example of MIT, the second part can be achieved by the syntax `site:mit.edu`. The university domains to consider were provided by us as a list of seed URLs.

The remaining thing is to choose query keywords that retrieve slide files exhaustively. One might think that it would be better *not* to specify a keyword to include *all* the slides in a university. If no keyword were specified, a large number of slide files would be matched with the query. Please note that Google does *not* show all the results but omit some low-ranked results to reduce the burden of their servers. This is reasonable because users are usually interested in only high-ranked results. According to our experience, it seems that Google retrieves no more than about 700 results per query. For example, in Fig. 6, we could access only about 700 results out

Table 1 The list of search keywords in alphabetical order

Aeronautics	Agricultural	American	Anthropology	Architecture
Area	Art	Art History	Astronomy	Bioengineering
Biology	Biomedical	Business	Cell	Chemical
Chemistry	Civil	Classics	Communication	Comparative
Computer	Conservation	Dance	Design	Ecology
Economics	Education	Electrical	English	Environmental
Ethnic	Evolution	Exercise	Film Studies	Foreign
Geography	Geology	Geophysics	Health	History
Industrial	Information	Journalism	Justice	Languages
Law	Library	Life	Linguistics	Literature
Marine	Material	Mathematics	Mechanical	Medicine
Meteorology	Molecular	Music	Natural	Neuroscience
Nuclear	Nursing	Nutrition	Operation	Philosophy
Physics	Plant Biology	Policy	Political	Psychology
Public	Religious	Resources	Social	Sociology
Statistics	Theater	Urban	Women	Zoology

of 2440 results returned. The implication here is that we need to reduce the number of query results by specifying a keyword with the `site` and `filetype` conditions.

Thus, our strategy was to run *multiple* queries using the keywords that would produce less than 700 results. Here, each keyword tends to be general, but should not be too general in order to avoid too many results. For example, a very general keyword "outline," which is likely to be included in most slides, is not suitable for our task. In addition, it is preferable to maintain a smaller number of search keywords to reduce the crawling time. For our development, we chose the keywords from the department or academic field names. Table 1 shows the list of 80 search keywords being used.

Putting these conditions together, we issue the queries of the form shown below. The `start=0` condition requests the first HTML page of the search result. In summary, per university, 80 queries were issued for `ppt` files and once more for `pptx` files.

http://www.google.com/search?q=*keyword*+filetype:ppt+site:*domain*&start=0

3.3.2 URL extraction

The URL extractor is responsible for retrieving the URLs of the slide files found by Google. It fetches the HTML pages of the search result using `GNU Wget`,[2] parses them, and looks for those URLs using regular expressions. Then, it removes the

[2] http://www.gnu.org/software/wget/.

duplicate URLs while combing the URLs.[3] Duplicate URLs can occur because a slide file can match with multiple search keywords in Table 1. The crawling procedure so far is summarized in Algorithm 1.

Algorithm 1 URL Crawling

INPUT: A set $\mathcal{I}$ of the seed URLs
INPUT: A set $\mathcal{K}$ of the search keywords (Table 1)
OUTPUT: A set $\mathcal{O}$ of the URLs of slide files
1: Initialize the output $\mathcal{O}$, i.e., $\mathcal{O} = \emptyset$;
2: **for** each $seed \in \mathcal{I}$ **do**
3: **for** each $filetype \in \{\texttt{ppt}, \texttt{pptx}\}$ **do**
4: **for** each $keyword \in \mathcal{K}$ **do**
5: Generate a query q_{curr} using $seed$, $filetype$, and $keyword$;
6: Issue q_{curr} to Google;
7: Fetch the first HTML page p_{curr} of the search result;
8: **repeat**
9: Extract the set $\mathcal{U}$ of the URLs on p_{curr};
10: /* Duplicate URLs are *not* accumulated here. */
11: Accumulate these URLs to $\mathcal{O}$, i.e., $\mathcal{O} = \mathcal{O} \cup \mathcal{U}$;
12: Sleep for one second or 15 minutes;
13: Fetch the next HTML page p_{next} of the search result;
14: $p_{curr} = p_{next}$; /* Proceed to the next page */
15: **until** p_{curr} is not empty
16: **end for**
17: **end for**
18: **end for**
19: Return the output $\mathcal{O}$;

Google blocks automated queries. To overcome this restriction, we have to sleep in between the requests for at least a second (Line 12). Empirically, we decided to wait one second after each and every request and to wait 15 min after every 150 requests. Because Google accumulates the number of requests from the same IP address, it is helpful to use multiple crawling machines (commodity PCs) to expedite this step. Running multiple threads within a single machine is not helpful at all owing to Google's blocking policy.

3.3.3 File downloading

The file downloader is a Java program of using the HTTP protocol to fetch the slide file pointed by a URL. Unlike the URL extractor, we prefer to fork many download threads as long as the network bandwidth is allowed. Each thread stores the slide files to an output directory along with their source URL.

Now, we would like to mention that our focused crawler is *polite*. Needless to say, if a crawler tries to download many files simultaneously per second from a single server, the server would have a hard time keeping up with a sudden increase of requests. Our crawler is *not* likely to send many requests to a single server *at*

[3] We understand that the same slide file can be located at many different URLs, and this type of duplication will have to be removed during indexing time after crawling.

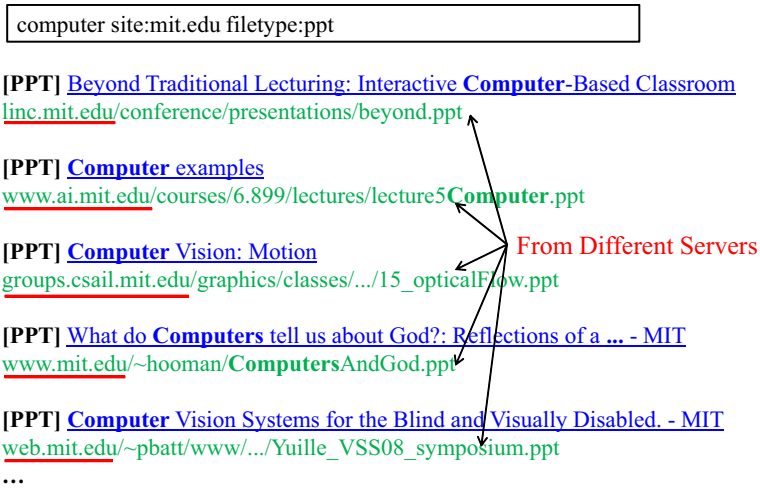


Fig. 7 An example showing the scattered URLs in Google search results

Table 2 The average number of new or updated slide files per university

Range	ppt	pptx	Total
Past 1 year	139.23	55.54	194.77
Past 1 month	11.60	4.62	16.22
Past 1 week	2.68	1.07	3.75

the same time. The slide files to download are actually sorted by their PageRank score, which is Google's ranking scheme, so they are *not* clustered by their URL as shown in Fig. 7. Naturally, the download requests are well scattered to the servers in a domain.

3.3.4 Incremental crawling

Incremental crawling, which refers to the process of crawling only new or updated contents, is easily implemented by our approach. Popular commercial search engines allow us to retrieve only the web pages or files *newly indexed during a given time range*. Such web pages or files have been newly indexed since they were either newly created or updated. Thus, just adding this time-range condition to a search query completes the implementation.

In order to determine the range, we counted the number of the slide files newly indexed into Google during the last 1 year, month, and week. Table 2 shows the average number of such files per university in the top-500 universities (as of May 2012). Based on this observation, we concluded that triggering incremental crawling *every week* adds only negligible overhead and, at the same time, makes the collection of our slide files almost always up-to-date.

Table 3 The list of universities and seed URLs for crawling

Rank	University	Seed URL
1	University of Cambridge	cam.ac.uk
2	Harvard University	harvard.edu
3	Yale University	yale.edu
4	UCL (University College London)	ucl.ac.uk
5	Massachusetts Institute of Technology (MIT)	mit.edu
6	University of Oxford	ox.ac.uk
7	Imperial College London	imperial.ac.uk
8	University of Chicago	uchicago.edu
9	California Institute of Technology (Caltech)	caltech.edu
10	Princeton University	princeton.edu
11	Columbia University	columbia.edu
12	University of Pennsylvania	upenn.edu
13	Stanford University	stanford.edu
14	Duke University	duke.edu
15	University of Michigan	umich.edu
16	Cornell University	cornell.edu
17	Johns Hopkins University	jhu.edu
18	ETH Zurich (Swiss Federal Institute of Technology)	ethz.ch
19	McGill University	mcgill.ca
20	Australian National University	anu.edu.au
...	...	...
500	Leibniz Universität Hannover	uni-hannover.de

Overall, we issue the queries shown below for incremental crawling. The condition `tbs=qdr:w` confines the time range to the last one week. Unlike Sect. 3.3.1, no search keyword is specified since the number of query results with the time-range condition is highly likely to be less than 700. Thus, only one query is executed for each domain every week.

http://www.google.com/search?`q=filetype:ppt+site:`*domain*`&start=0&`**`tbs=qdr:w`**

4 Execution results

4.1 Seed URLs

Because the goal of our vertical search engine CourseShare is to collect slide files from academic institutions, we chose the top-500 universities for crawling targets

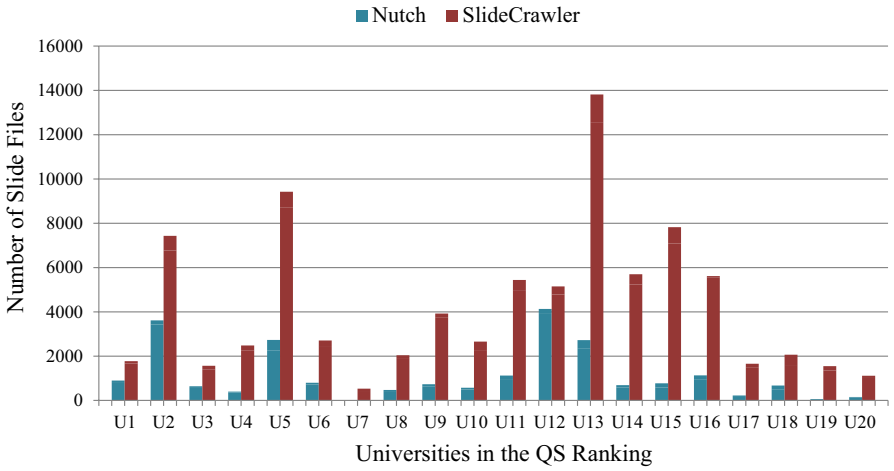


Fig. 8 Comparison between Nutch and SlideCrawler for the top-20 universities

based on the QS World University Rankings[4] in 2010. Part of the universities and seed URLs is shown in Table 3.

4.2 Comparison with Nutch

To examine the effectiveness of the proposed approach, two approaches below were compared for the number of slide files crawled.

- SlideCrawler: our focused crawler.
- Nutch: The web pages were crawled using Nutch, and then, the URLs of slide files were extracted from the web pages crawled. The same set of seed URLs in Table 3 was provided to Nutch. The crawling depth was set to be 7, which was enough.

Figure 8 shows the number of slide files crawled by the two approaches from the top-20 universities. As shown, SlideCrawler outperformed Nutch at every university site examined, without exception. In total, SlideCrawler crawled 84,430 slide files, and Nutch 22,523 slide files. That is, SlideCrawler collected 3.7 times more slide files than Nutch. SlideCrawler is free from the limitations explained in Sect. 3.1 whereas Nutch is not. This is the main reason why we can observe this big improvement.

Interestingly, Nutch could not crawl any files from U7 (Imperial College London) because of the robots exclusion protocol. The "robot_denied(18)" error occurred when Nutch tried to connect to the university site. This absence of results directly shows the limitations of this approach.

[4] http://www.topuniversities.com/university-rankings/world-university-rankings/.

Table 4 The number of slide files crawled from the top-500 universities

Rank	Number of `ppt` files	Number of `pptx` files
1	1664	105
2	6772	655
3	1402	165
4	2266	217
5	8726	695
6	2459	247
7	381	152
8	1874	167
9	3730	194
10	2273	383
11	4982	458
12	4777	371
13	12,562	1248
14	5236	459
15	7078	743
16	5541	72
17	1470	184
18	1598	467
19	1344	203
20	1053	57
21–100	232,134	27,126
101–200	174,073	15,963
201–300	121,742	12,439
301–400	86,258	9255
401–500	80,648	9664
Total	772,043	81,689

Table 5 The execution time of SlideCrawler for the top-20 universities

	URL crawling	File downloading
Execution time	45 h 39 min	4 h 51 min

4.3 Detailed results

4.3.1 Number of files crawled

Table 4 shows the detailed crawling result of SlideCrawler. For the top-500 universities, 772,043 `ppt` files and 81,689 `pptx` files were collected; thus, in total, 853,732 slide files were collected. We did our crawling in the fall of 2011. This number is expected to increase if we add more search keywords or seed URLs.

4.3.2 Execution time

We also measured the execution time spent for crawling slide files from the top-20 universities. Only 20 universities were considered here because it is hard to precisely measure the execution time for a long-running task owing to periodic system maintenance. Table 5 reports the execution time, where URL crawling (Algorithm 1) includes the first two steps in Sect. 3.3. For this experiment, four commodity PCs were used for URL crawling, and one commodity PC for file downloading.

The four PCs processed different seed URLs in parallel, but only one thread was executed within each PC to avoid our crawling machines being blocked by Google. In contrast, one hundred threads were executed to download the slide files collected from the previous steps. That is, file downloading can exploit *intra*-machine parallelism whereas URL crawling cannot. Thus, it would be better to allocate more machines to URL crawling than to file downloading.

Overall, we believe the performance of our focused crawler is acceptable considering that it can collect about 50,000 slide files using only five commodity PCs per day.

5 Conclusions

In this paper, we have proposed a new approach to augmenting a focused crawler that collects academic slides. Our focused crawler, *SlideCrawler*, takes advantages of the advanced features of a general web search engine, reducing implementation effort and achieving satisfactory performance. More specifically, our focused crawler consists of only 6000 lines of Java codes and is shown to collect several times more slide files than Nutch. Though one might argue that our approach is not a self-contained solution, we contend that our approach is very cost-efficient and effective. Moreover, our research is a timely topic since vertical search engines are poised to become more and more popular in this era of Big Data. Overall, we believe that we have provided a practical and useful approach to enhancing a focused crawler exploiting the power of general web search engines.

Acknowledgements This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. 2017R1E1A1A01075927).

References

1. Boldi P, Codenotti B, Santini M, Vigna S (2004) UbiCrawler: a scalable fully distributed web crawler. Softw Pract Exp 34(8):711–726
2. Bonato A, del Río-Chanona RM, MacRury C, Nicolaidis J, Pérez-Giménez X, Prałat P, Ternovsky K (2018) The robot crawler graph process. Discrete Appl Math 247:23–36
3. Boukadi K, Rekik M, Rekik M, Ben-Abdallah H (2018) FC4CD: a new SOA-based focused crawler for cloud service discovery. Computing 100(10):1081–1107

4. Chakrabarti S, van den Berg M, Dom B (1999) Focused crawling: a new approach to topic-specific web resource discovery. Comput Netw 31(11–16):1623–1640
5. Chakrabarti S, Punera K, Subramanyam M (2002) Accelerated focused crawling through online relevance feedback. In: Proceedings of 11th International World Wide Web Conference, Honolulu, Hawaii, pp 148–159
6. Chau M, Chen H (2003) Comparison of three vertical search spiders. IEEE Comput 36(5):56–62
7. Cho J, Garcia-Molina H (2000) The evolution of the web and implications for an incremental crawler. In: Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt, pp 200–209
8. Cho J, Garcia-Molina H (2000) Synchronizing a database to improve freshness. In: Proceedings of 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, pp 117–128
9. Dean J, Ghemawat S (2004) MapReduce: simplified data processing on large clusters. In: Proceedings of 6th Symposium on Operating System Design and Implementation, San Francisco, California, pp 137–150
10. Diligenti M, Coetzee F, Lawrence S, Giles CL, Gori M (2000) Focused crawling using context graphs. In: Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt, pp 527–534
11. Edwards J, McCurley KS, Tomlin JA (2001) An adaptive model for optimizing performance of an incremental web crawler. In: Proceedings 10th International World Wide Web Conference, Hong Kong, China, pp 106–113
12. Gantz J, Reinsel D (2012) The digital universe in 2020: bigger digital shadows, and biggest growth in the far east. Technical Report, IDC
13. Heydon A, Najork M (1999) Mercator: a scalable, extensible web crawler. World Wide Web 2(4):219–229
14. Kleinberg JM (2001) Small-world phenomena and the dynamics of information. In: Proceedings of Advances in Neural Information Processing Systems, vol 14, Vancouver, British Columbia, pp 431–438
15. Koster M (2018) A standard for robot exclusion. http://www.robotstxt.org/orig.html. Accessed on 07 Jan 2018
16. Kunder M (2018) The size of the world wide web (the internet). http://www.worldwidewebsize.com/. Accessed on 07 Jan 2018
17. Langville AN, Meyer CD (2006) Google's PageRank and beyond: the science of search engine rankings. Princeton University Press, Princeton
18. Lee W, Leung CKS, Lee JJH (2011) Mobile web navigation in digital ecosystems using rooted directed trees. IEEE Trans Ind Electron 58(6):2154–2162
19. Menczer F, Pant G, Srinivasan P (2004) Topical web crawlers: evaluating adaptive algorithms. ACM Trans Internet Technol 4(4):378–419
20. Pal A, Tomar DS, Shrivastava S (2009) Effective focused crawling based on content and link structure analysis. Int J Comput Sci Inf Secur 2(1):80
21. Pant G, Srinivasan P, Menczer F (2004) Crawling the web. In: Poulovassilis A, Levene M (eds) Web dynamics. Springer, Berlin, pp 153–178
22. Pirkola A (2007) Focused crawling: a means to acquire biological data from the web. In: Proceedings of VLDB workshop on data mining in bioinformatics, Austria, Vienna
23. Shemshadi A, Sheng QZ, Qin Y (2016) ThingSeek: a crawler and search engine for the internet of things. In: Proceedings of 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, pp 1149–1152
24. Shkapenyuk V, Suel T (2002) Design and implementation of a high-performance distributed web crawler. In: Proceedings of 18th International Conference on Data Engineering, San Jose, California, pp 357–368
25. Tatli EI, Urgun B (2017) WIVET-benchmarking coverage qualities of web crawlers. Comput J 60(4):555–572
26. Vieira K, Barbosa L, da Silva AS, Freire J, Moura E (2016) Finding seeds to bootstrap focused crawlers. World Wide Web 19(3):449–474
27. Wikipedia (2018) Focused crawler. http://en.wikipedia.org/wiki/Focused_crawler. Accessed on 07 Jan 2018
28. Wikipedia (2018) Vertical search. http://en.wikipedia.org/wiki/Vertical_search. Accessed on 07 Jan 2018

29. Yin C, Liu J, Yang C, Zhang H (2009) A novel method for crawler in domain-specific search. J Comput Inf Syst 5(6):1749–1755
30. Zhao F, Zhou J, Nie C, Huang H, Jin H (2016) SmartCrawler: a two-stage crawler for efficiently harvesting deep-web interfaces. IEEE Trans Serv Comput 9(4):608–620