Utrecht University

A Web Crawler for Automated Document Retrieval in Health Policy

Master's Thesis: Applied Data Science
INFOMTADS

Michelle Donovan
7204345

Project Supervisor: Prof. Yannis Velegrakis
Second Supervisor: Dr. Rick Vreman
Progress Coordinator: Dr. Mel Chekol

July 5, 2021

Contents

Abstract

Document retrieval in Health Policy Research is labor-intensive and inefficient. To investigate the efficacy and transparency of health policy processes such as drug approval, reports are manually collected from the websites of health regulatory bodies. This paper discusses the configuration of a web crawler to automate this process. The usage of Apache Nutch to crawl the European Medicines Agency (EMA) and retrieve European Public Assessment Reports (EPAR) is detailed. The crawler is designed to be successful in the context of EMA but Nutch provides capabilities for wider applications which are also documented. The crawler was successful in gathering the correct URLs creating a database of the target reports. The scalability of this web crawler is apparent in terms of the Nutch capabilities however, some of the configurations remain context specific. The extensible nature of the crawler properties, although valuable, require extensive knowledge to implement. This paper provides a detailed description of how to crawl EMA and provides guidance on how this configuration can be applied to other contexts. Future research into the range of Nutch capabilities is recommended to ensure the tool is being used to full capacity.

1 Introduction

1.1 Overview & Relevance

Large-scale retrieval of information online such as documentation and records has become a routine part of the academic research process. Researchers require easy, efficient access to publications to evaluate current research processes and to utilize the mass of information that is available to them [1].

Document retrieval is an information science term for acquiring a set of documents with specific characteristics from a larger data set, along with information about their relevance [2]. Research papers are subject to systematic reviews to create an overview of research on a particular topic and to evaluate new research avenues [1]. These reviews utilize academic databases and search engines. PubMed is an example of a database where keyword searches can result in document retrieval to form a collection of particular medical documents [3].

Document retrieval in domains such as medicine remains time consuming, labor intensive and expensive [1]. Information becomes quickly outdated [4]. These challenges are amplified when research pertains to non-academic document retrieval. There is a lack of structured resources to draw from, particularly in health policy research [5]. The required content lives in resources that are not designed for mass document retrieval [6]. Common systematic review methods such as PRISMA have been reframed and applied to health policy contexts and include additional manual searching and individual document downloading [7]. There are cases documenting the manual retrieval of health records to create a health record database that can be accessed by others [8, 9].

1.2 Document Retrieval in EMA

One health policy context currently challenged by inefficient document retrieval is the assessment of drug approval procedures. Organisations such as the European Medicines Agency (EMA) are involved in assessing drugs in terms of their safety. HTA-bodies (Health Technology Assessment) weigh the economic benefits and risks of certain drugs. The EMA provides guidance to the HTA bodies however, their recommendations vary depending on the type of research methods employed and the regulatory processes involved [10, 11]. Drug approval may be accelerated if technologies are seen to have urgent added value and, requirements such as controlled randomized studies are sometimes bypassed [10, 11].

There are disparities between the drug regulators who address health risk and the HTA-bodies who assess economic value [12]. There are also disparities within regulatory bodies like EMA [10, 11]. Regulatory bodies publish a series of reports, documenting their assessments which can be analysed to evaluate these issues. It is vital that these organisations are consistent and efficient to ensure that health technology reaches the market quickly but safely.

In order to assess the processes in which organisational bodies review health technology, as well as how different types of research are assessed within one organisational body, documents such as drug assessment reports must be compiled and analysed. Consistent and efficient approval procedures are necessary to ensure that the market remains up to date with industry and that people are getting the best available care [11].

Documents from health organisations like EMA are open access and facilitate scientific exploration. Despite liberal permission requirements, this type of resource collection faces limitations. Manual collection of each individual document from a series of web page links is required. There can be a lack of available documents at the time of collection and documents are changed or reviewed at least annually [11]. Furthermore, collected data cannot be applied to any other health regulatory body [11].

This process of report retrieval is time consuming and labor intensive. The goal of this research is to automate the document retrieval process. We create a tool that automates the retrieval of public assessment reports from EMA using web crawling techniques. We create a blueprint of a web crawler with Apache Nutch within this context. The web crawler aims to be widely applicable so that various document types and organisations can be investigated using similar methods.

1.3 Problem Approach: Web Crawling

There have been efforts to create automated document retrieval within health policy. Marshall et al. (2020) created a continuously updating database of clinical trial reports using retrieval from a structured database [13]. Other research in automating document retrieval in the scientific context also focuses on utilizing existing databases effectively [14, 15]. The challenges facing health policy relate to an earlier stage of the process where the database of documents must also be compiled.

Web crawling has been used to overcome this obstacle in the biomedical domain and has the advantage of utilizing real time information [16]. Web crawling is designed for information retrieval from the web [17]. Crawling uses the hyperlink structure of the internet to "crawl" from one URL to another, collecting the content of each web page it visits [18].

A sample of the EMA website structure is shown in Figure 1. The EMA website has a front page which is made up of HTML. HTML is the standard language of websites and contains information about the design and formatting of a web page as well as the text, images and hyperlinks embedded in the page. Hyperlinks allow a user to navigate from one location within a website to another, as well as to other web domains. Web crawlers operate on the hyperlink structure of web pages. Seven of the links on the EMA front page are shown in Figure 1. Once a link on the home page is clicked, such as "Medicines", a new web page is shown which contains content such as text and a new set of links. In this case, if the "Downloads" link is then followed, a page with information regarding available documents on EMA is displayed.

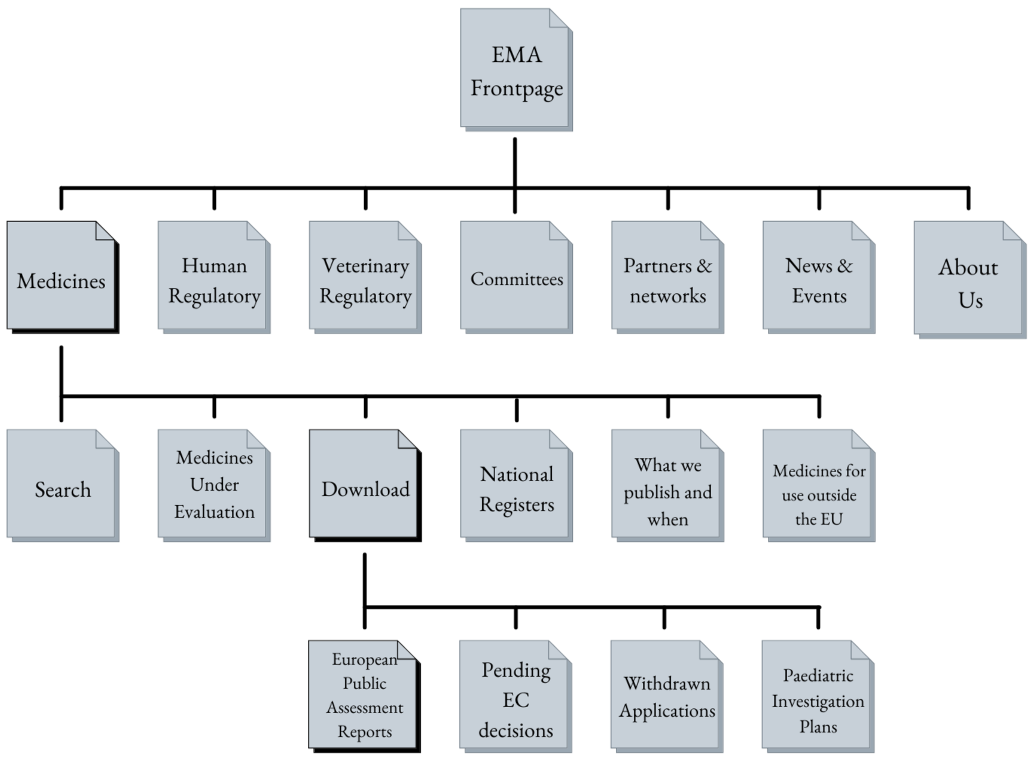

Figure 1: Website Architecture Example: EMA

A web crawler or spider begins at a source URL set by the user, such as the EMA front page. The crawler collects all information on the initial page, including each link. The links in this example include "Medicines" and "About Us" and are stored in URL format. Once all information is gathered from the source URL, this data is stored and indexed which allows easy retrieval of the data. The crawler then follows every link it has previously stored. These links relate to the second level of Figure 1. The crawler stores all information on each of the seven web pages and indexes them.

A "hop" refers to the action of the crawler jumping from one level of a web site to a deeper one. A full crawl of the architecture shown in Figure 1 would require four hops, or a "depth" of 4. However, websites are much more vast, complex and interconnected than the diagram shown. Search engines such as Google are well indexed web crawlers. For a search results to be returned by Google, it must be previously crawled and indexed. The hierarchy of the search results is determined by the relevance to the search query and also how many in-links and out-links the web page has. The number of in-links is determined by how many other web pages contain a link to the target web page. Out-links refer to links are contained on the target web page, such as "Medicines" is contained in the EMA front page, as shown in Figure 1.

To understand the challenges of presenting a tool with the necessary capabilities and to highlight the suitability of a web crawler in this context, an overview of the manual process which we are trying to automate is shown for the case of a cancer medicine, Keytruda in Figure 2. There is a similar process for retrieving documents from other health agencies such as the National Institute for Health and Care Excellence (NICE) and Zorginstituut Nederland (ZIN).

The process begins with a keyword search. This elicits search results which are scanned for the European Public Assessment Report web page (EPAR). This page contains all information pertaining to the drug assessments conducted by EMA. There are various reports available on this page. The target one in this case is titled "Keytruda: EPAR - Public assessment report". Clicking on this link will prompt download options for the PDF file.

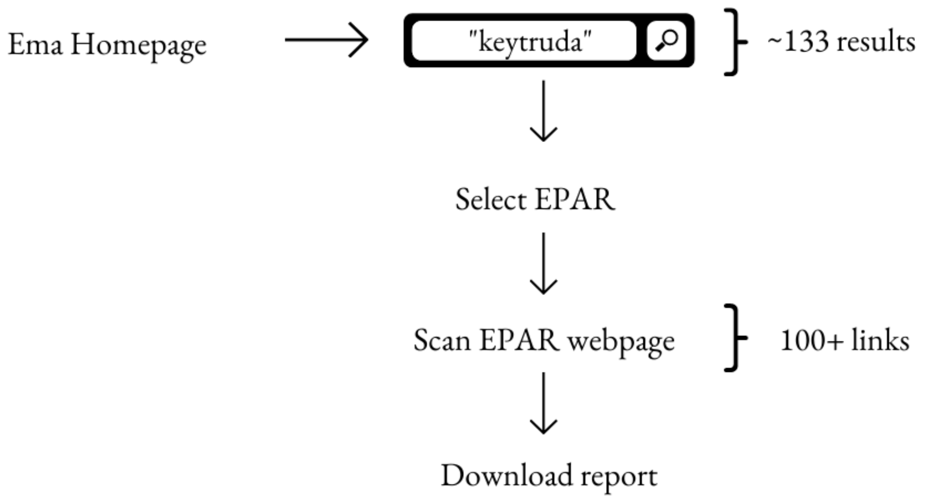

Figure 2: Manual Collection of EPAR for Keytruda

A number of the limitations to the manual process of document retrieval from EMA described previously are overcome with the use of a web crawler. These limitations translate into the required tool features of a web crawler, as is shown in Table 1. It is clear from this table that the limitations of document retrieval in the health policy domain translate directly into required tool features.

The features listed in Table 4 are key components of web crawlers. Web crawlers facilitate automated document retrieval and once configured, are time efficient. The manual retrieval of documents is sensitive to human error whereas web crawlers, although subject to error, allow tracking of errors and provide information on any web pages that were not crawled successfully. The manual updating of a database is time consuming and requires engagement with the PDF content. It is not always clear if changes have been made. Web crawlers gather HTML which contains clear information that is often not human readable regarding the most recent updates. It can then download the updated content. Finally, manual gathering of documents lead to research conclusions that can only be applied to the organisational bodies that were reviewed. Web crawlers contain configuration options that can be adapted to different websites without previous work being redundant.

Table 1: Translation of Manual Limitations to Automation Features

Limitations	Features
Labor Intensive	Automatic
Time consuming	Efficient
Subject to human error	Reliable
Difficult to track errors	Error tracking & traceability
Unpredictable availability	Recursively updated database
Document updates are difficult to track	Tracked updates
Context dependent & not scalable	Context independent & scalable

1.4 Apache Nutch

There are many available web crawlers with a range of capabilities and Apache Nutch was chosen for this project. It has vast documentation and is open source. It has an active support network for troubleshooting [19]. A review of these characteristics in web crawlers developed with Java are shown in Figure **??** [19] in Appendix A and Nutch ranks highly compared to others.

A comparison of open source crawlers was completed in 2015 between Scrapy, Apache Nutch, Heritrix, WebSphinix, JSpider, GnuWget, WIRE, Pavuk, Teleport, WebCopier Pro, Web2disk and WebHTTrack on characteristics such as quality, licensing, operating systems, politeness, distritubiton, types of links visited and scalability [20]. Apache Nutch performed well in multiple categories, particularly in terms of scalability due to it's use of Hadoop data structures[1] [20]. Nutch has a comprehensive list of politeness policies. These ensure a web crawler obeys rules set by the web designer, such as limitations on which pages can be crawled and how regularly. Adequate politeness policies reduce the chance of overloading a website and becoming blacklisted [21, 20].

Web crawlers are used for website testing, model checking and assessing security risks, as well as information retrieval [21]. Crawlers have different design features that are suitable to certain tasks. For example Scrapy is an extremely fast tool but is limited in scalability [20]. Apache Nutch is highly scalable and performs well in terms of quality [20]. The modularity of Nutch and the range of plugins[2] available such as those for parsing[3] ensure flexible configuration [22, 23]. The software is developed by Apache therefore integrates seamlessly with Apache indexing tools such as Solr and Elastic Search [4]. The extensibility of Nutch in particular allows the wide application of this crawler to different drug approval websites and health policy contexts.

[1]Hadoop is a framework for storing massive amounts of structured and unstructured data in a distributed manner

[2]A plugin is a separate piece of software that can be added to a program to improve capabilities

[3]Parsing is converting raw code to readable data, such as converting the text content on a HTML to entries in a database

[4]Solr and Elastic Search are indexing tools that facilitate search operations and filtering results

1.5 Research Aims

This paper details the configuration of Nutch for document retrieval from EMA. Automating this process will significantly reduce the time needed to investigate the drug approval processes in place and provide a framework for web crawling in wider applications across health policy research. The capabilities and performance of the crawler for document retrieval are reported as well as the challenges faced and suggestions for future directions.

The features shown in Table 1 highlight the components that are required to make a data science tool to solve the issue of efficient document retrieval in health policy. In order to assess whether the crawler has successfully achieved the goals of this project, these features are summarised into three assessment criteria; document selection(accuracy), document retrieval(completeness), and scalability.

2 Methods

2.1 Set-up & Materials

The software used to run Nutch is shown in Table 2. Nutch was installed and run in Windows 10(64-bit), on a laptop with an Intel Core i7 processor running at 2.6GHz, using 8GB of RAM. Cygwin is used to simulate a UNIX environment on Windows, as is required for Nutch. Nutch utilizes Hadoop data structures to facilitate clustering. A cluster is where the same process can be run on different connected machines to maximize performance and reduce run-time. To integrate Hadoop into Nutch on Windows, files from a Github Repository are downloaded and added to the bin folder in the Hadoop installation directory. Java and Hadoop are set up as environment variables[5]. Apache Solr is used to provide a search ability to the data collected by Nutch. A Nutch core is set up in Solr which incorporates all meta-data about the Nutch data so that it can be queried effectively. Solr is run through the Windows Command Prompt as it is not compatible with Cygwin. More details on Nutch set-up can be found on the Apache Nutch Wiki [24].

Table 2: Software Requirements

Software	Version
Apache Nutch	1.18
Apache Hadoop	3.3.0
Java (JRE)	1.8
Apache Solr	8.5.1

[5]Environment variables are only required for a Windows set-up.

Nutch is run through the UNIX command shell, using Bash[6]. Figure 3 displays a selection of the most utilised files in the Nutch installation directory and their structure. There is a "crawl" script, located in the bin directory that compiles all commands required for a complete crawl. Commands can be run individually using the "nutch" script. A detailed description of each Nutch command and it's usage can be found in the online manual [25].

There is a "conf" folder which holds XML and text documents that can be edited to apply specifications to Nutch. These include scripts to implement politeness policies, plugins as well as crawl restrictions. The "nutch-default" file describes all default settings in Nutch. Examples of these properties are shown in Table 5 in the Appendix. These settings are altered using the "nutch-site" file which overwrites the default file. The "regex-urlfilter" is used to restrict which URLs are crawled. Data from each Nutch crawl is stored in the "crawl" folder . The terminal history for each crawl is saved in daily Hadoop log files. The Python IDE, Spyder and the Windows native text editor, Notepad are used to read and edit the configuration files and crawl output. Source URLS are added to the "seed.txt" in the URLs folder.

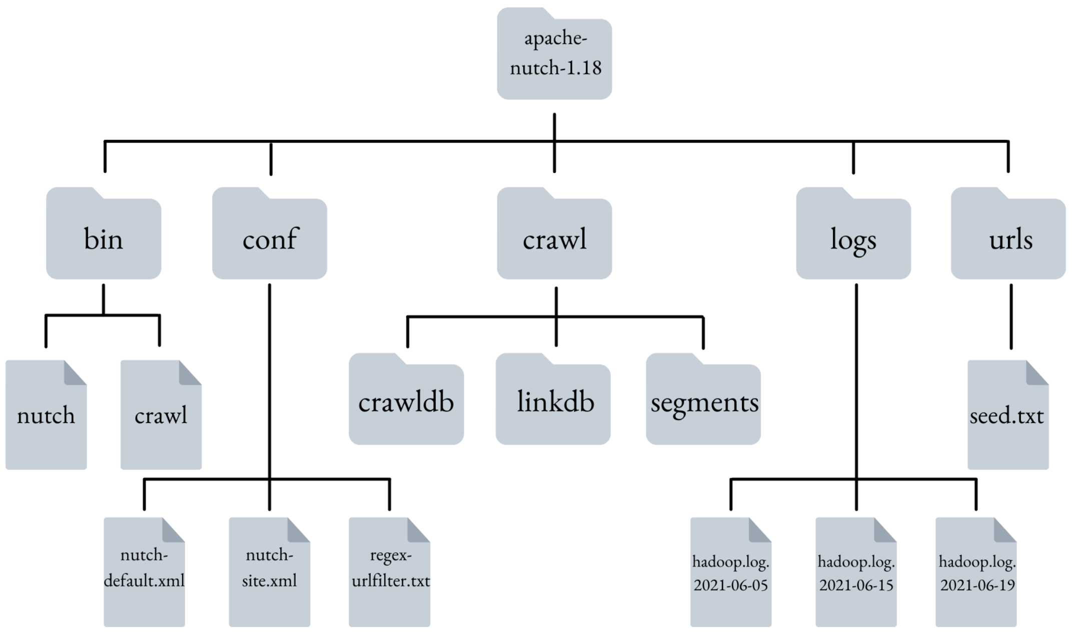

Figure 3: Nutch File Structure

2.2 Nutch Process

The crawl script runs six commands, as shown in Figure 4. The first step is to inject the source URL(s) specified in the seed.txt into the database in Nutch, the "crawldb". The source URL is where the crawler begins. A "segment" is then generated which means a directory is created and labelled with the current timestamp. This folder will contain all the data downloaded from this crawl run, including text and other content. Segments allow the division of information from different crawl

[6]Bash is the coding language used in the UNIX environment.

runs and iterations as the crawldb does not record this and will hold all data until cleared. They are also translated into Solr. A queue of all the URLs to crawl is generated. In this step, regular expressions or regex are used for URL string matching. The regex-urlfilter is used to specify which URLS can be crawled and imposes navigation rules to the crawler, such as to remain within the domain of EMA or to only crawl PDFs.

"Fetching" refers to the downloading of raw data from the web pages in the crawl database. On a first run or iteration, the only URLs in the crawldb are the web pages that have been added to the seed.txt. The raw data from the web pages is then parsed and the text content is extracted. Each link contained on the seed URLs is parsed. Nutch has plugins to parse various data types such as images and videos but only text and other hyperlinks were of interest in this case.

The databases in Nutch are then updated with all new information. The parsed text is stored in the segments directory and all of the new links found on the seed URLs are added to the crawldb. The crawldb contains all metadata from the crawl, including URLs and their fetch time and status, meaning whether a page was fetched or not. The database is also cleaned of duplicate records. Web page signatures are stored so Nutch can keep track of updates on the same URL. If a web page changes in content, a new record will be created with the same URL and a new signature.

The scores of each web page are stored in the crawldb and is one of they key features of Nutch. This score represents how many in-links there are for a particular URL. For example, a link to the EMA frontpage is likely to be present on health policy and government websites therefore, it would have a high score in comparison to a EPAR PDF file that is likely only available on the EMA website. This feature is not relevant for this application of Nutch but is an important capability of the crawler.

The "linkdb" is updated with information regarding the anchors of the links, meaning the text they are connected to on the site. This database also provides information on in-links and out-links. The databases are then indexed using Solr.

This process is repeated according to the depth parameter set. If the depth is set to two or more, the process will begin again but after the first crawl round is indexed, the process restarts at the generate command. The newly found URLs, as well as the source URLs are now contained in the crawldb. Each URL is added again to the queue and a second segment is generated with a new timestamp.

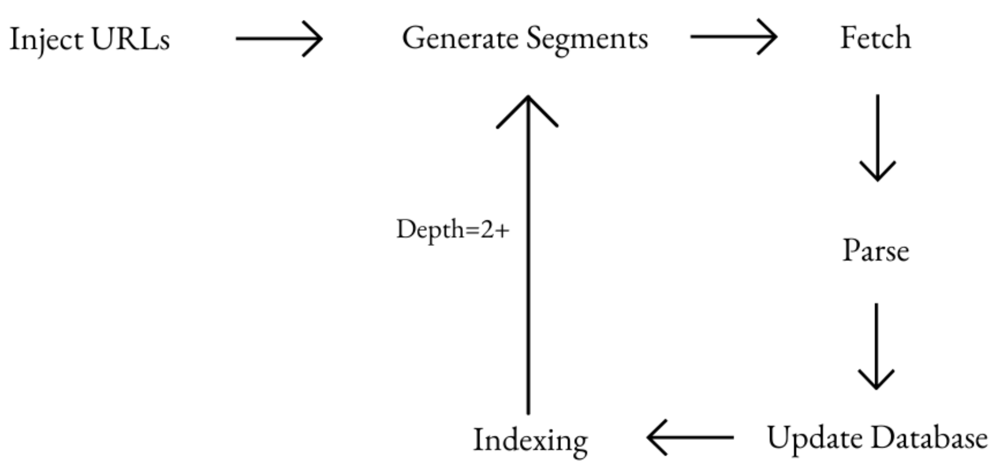

Figure 4: Crawl Process

2.3 Configuration of Nutch

The first approach to the Nutch configuration is to determine an appropriate source URL and apply a regex filter. This gives Nutch the ability to begin at a specific location and crawl only selected web pages rather than all links on a page. The filter can also be applied to restrict the crawler to a domain. A domain specific crawl can also be implemented using the "db.ignore.external.links" property, shown in Table 5 in the appendix. There is a suffix-urlfilter which limits the crawl to certain file types, such as PDFs.

Target documents such as the EPAR reports follow a specific URL pattern. The regex-urlfilter is edited to allow URLs to remain in the domain of EMA and also to ensure only documents related to EPAR were crawled, as shown in Figure 5. This regex is implemented in a crawl of depth 3 run from the EMA search page with EPAR search filters applied. The search filters appear on the web page as tick-able boxes and were refined to only include EPAR related search result. A crawl of depth 3 was run from the search page with no search filters with the EPAR regex filter implemented also. A crawl of depth 4, 5 and 6 were run from the EMA front page with only a domain specific regex filter.

```
+^(http|https)://www.ema.europa.eu/((?i).*epar.*)
```

Figure 5: Regular Expression

The second approach is to compile a list of initial URLs to ensure the target URLs are crawled at a specific depth. An exploration of the target website is required to evaluate which source URLs may be successful in navigating to the target document. Websites may have sitemaps[7] which can help explore potential source URLs or there may be archive lists or overviews of the documents published on a health policy website. EMA provides an excel file listing all the EPAR web pages for each drug they have assessed. These web pages are not the reports themselves but contain all information and documents relating to the assessment, including a link to download the target EPAR document. This excel file provides an ideal seed.txt. The URL for each drug EPAR web page is added to seed.txt and a crawl is run at a depth of 1. Filters are not necessary here as the correct URLs have already been selected.

Nutch downloads the content from each web page and stores it in binary form. The data is then parsed and out-links and text are extracted. The aim for this crawler is to provide a database of documents which can be used for analysis and exploration therefore, the human readable, PDF format of the document is required. Nutch does not provide this therefore, the crawler compiles a list of the necessary URLs which can be downloaded externally.

Properties are added to the nutch-site file to overwrite the default settings. These properties can be seen in Table 3. All other recommended settings are appropriate for this crawl. The first property in the table names the crawler which follows crawling etiquette. The fetch interval property is set to 0 to allow for multiple crawl trials and consistently monitors pages for updated content. The maximum number of redirects ensures that if a page has been moved, the crawler will follow the new URL. The default value for this is 0, which resulted in many EMA pages not being fetched. The fourth property ensures that all links on the EPAR web page are stored. EMA provides documents in multiple languages therefore the default limit of 100 was exceeded before the EPAR report could be collected. A selection of the default properties can be seen in Table 5.

[7]A sitemap is a list of all accessible web pages on the domain and can be displayed by adding "/sitemap.xml" to the end of the domain URL

Table 3: Configuration of Nutch Properties

Nutch Property Name	Value	Description
http.agent.name	HealthPolicyCrawlerUU	
db.fetch.interval.default	0	The default number of seconds between re-fetches of a page (30 days).
http.redirect.max	5	The maximum number of redirects the fetcher will follow when trying to fetch a page. If set to negative or 0, fetcher won't immediately follow redirected URLs, instead it will record them for later fetching.
db.max.outlinks .per.page	-1	The maximum number of outlinks that we'll process for a page. If this value is nonnegative (>=0), at most db.max.outlinks.per.page outlinks will be processed for a page; otherwise, all outlinks will be processed.

2.4 Data Output

Data in Nutch is output into three databases in the crawl folder, as shown in Figure 1. The linkdb and crawldb databases are stored in a truncated format and a command is required to transform them into csv format. The segment content can also be outputted and can be explored with a text editor.We only focus on the URLs gathered in this context and all of the links found on each page is added to the crawldb so this is the only relevant database in this case.

The crawldb was cleaned using Python and only the URLs which string matched "epar-public-assessment-report" were kept as this was contained in the URL of the target documents. Once all relevant URLs were subset, the Wget download function was used to acquire all the target PDFs and these were stored locally. The python script used and some crawl demo's are available on Github.

3 Results

3.1 Nutch Capabilities

Crawls were run from the search page implementing the regex filter and a whole domain crawl was also run. It was quickly realised that EMA does not permit crawls of the search query page and as HTTP 403 errors were produced when crawling which communicate that requested access is forbidden. The search query is the only way to navigate to the EPAR reports from any web page on the EMA domain therefore a crawl of the whole domain at depths up to 6 does not ever gather the target documents. Other health policy websites such as ZIN (Zorginstituut Nederland) do not have these crawling restrictions therefore similar crawl configurations may be successful elsewhere.

EMA crawls with the first approach highlighted that Nutch capabilities are limited by dynamic web pages and search query APIs. For example even if search queries are not restricted as on EMA, most search results cannot be crawled. On the EMA search results page there is a "Load More" button. Once a web site user clicks this button, more search results are fetched. This means a web crawler can only access a limited amount of resources on this page as the links are not embedded as outlinks. Other web domains have pages of search results which cause similar difficulties for web crawlers.

Crawl attempts implementing regex filters can lead to challenges. The specified filter only accepts URLs that contain "EPAR" which means the crawler can only begin at a URL that fits this filter. It is difficult to begin at the "Medicines" page in EMA and then focus on EPAR reports. Similarly with the implementation of a PDF suffix filter, we cannot only include PDF files as this would require a PDF as a starting point for the crawl. Due to these restrictions, a list of URLs that either contain the necessary content or lead to it, is required to retrieve documents from EMA and likely other similar websites.

3.2 Performance Testing

The crawl configuration utilizing the 1817 source URLs provided by the spreadsheet overview of EPAR reports on EMA and was run at a depth of 1. To test the performance, this crawl configuration was run seven times. Crawl results are displayed in Table 4. For each crawl run, a new segment directory is created and the existing crawldb is updated. To ensure reliable results of the total number of links crawled each run and an accurate crawl time estimation, the database and indexes were moved to a test folder after each crawl. A clean crawldb is then generated and we can also adequately assess how long the entire process takes. When utilizing Nutch, this is not necessary as the database of previously crawled links is useful, it was done here for performance testing only.

The seed.txt contained 1817 URLs. Each of these URLs represent the EPAR web page for each drug assessed by EMA. After these URLs were injected, no URL was rejected by filters and all URLs were successfully fetched. On average 53% of these URLs were parsed and indexed in Solr. Over 62,000 out-links were fetched from the 1817 source URLs. The crawl duration stayed stable however these results may vary depending on bandwidth and connection stability.

Table 4: Crawl Results

Crawl no.	Duration (hr:min:sec)	Documents Indexed	Links stored
1	03:04:24	981	74854
2	03:01:29	981	74832
3	03:01:21	826	62894
4	03:01:22	971	74283
5	03:02:14	978	74480
6	03:01:45	988	75315
7	03:01:47	1001	76285

4 Discussion

4.1 Document Selection & Retrieval

The results of the EMA crawls are discussed first in relation to document selection (accuracy) and document retrieval(completeness), followed by a reflection on the scalability of Nutch in relation to the capabilities mentioned. Document selection is made significantly less complex when a list of useful starting points such as the excel file provided by EMA is utilised. There is no requirement of regular expressions as the crawls began at specified locations. This method is accurate, particularly if health organisations include all assessment information in such a spreadsheet. However, if web pages are omitted this method becomes significantly less accurate as it is difficult to know which documents have been missed. It is likely that these overviews are not complete or regularly updated. When utilised, the regular expression filters are effective in filtering out unwanted URLs however extensive knowledge of the web site architecture is required to ensure this method of document selection is accurate.

Document retrieval of the EPAR web pages from the EMA excel file was not as complete as expected. All URLs were fetched but only half were successfully parsed out of 1817. The reports for many of the drugs were missing. There are a range of parse plugins available for Nutch and the Tika Parser was used in this case which has a range of capabilities. Experimentation with the various parse options and different plugins may lead to improved performance of extracting links from these pages as there was no issue with the fetch process.

Documents were downloaded externally and all URLs that matched the target sub-string, "epar-public-assessment-report" were retrieved in PDF format successfully. The use of Wget facilitates open decision making in the crawling process as researchers are free to choose various file formats. The text content of each web page is stored in Nutch and can be queried easily with Solr which has a user interface. PDFs were required in this project, but if the goal of research is an NLP like task, the raw text content is already provided by Nutch for data analysis. The text content of the documents is also retrieved successfully.

4.2 Scalability & Wider Applications

Apache Nutch itself is highly scalable however there are aspects of this configuration that are limited to the EMA context. This crawl is generated from a seed.txt that is supplied by EMA. If domains provide a similar list, this can be used as the source of the crawler. However, this may not be the case for all websites. A crawl of the EMA domain was attempted using depths up to 6 and no regex filters however, most of the results are fetched using APIs and dynamic web pages therefore it is not possible to gather the PDF documents in the case of EMA regardless of depth settings. This may be an issue for crawling other health policy websites. If there are archive lists of the documents on the website, this is very useful.

Nutch is modular which means that software can be easily developed and integrated to overcome challenges in the crawl process. Web crawling is a commonly used tool that has already been scaled massively which creates a vast amount of information on how to scale Nutch. The software used is owned by Apache therefore if data architecture changes, Nutch is likely to be updated accordingly meaning the crawler properties will not become redundant.

Running a crawl does not take a lot of previous coding background to run, but rather an understanding of the Nutch process, as is provided in this paper. This ensures it can be operated by health policy researchers from non-technical backgrounds. The configuration of Nutch is complex as it requires extensive knowledge of the available properties and how they interact with one another. However, once configuration is complete, the commands are straightforward. They are run from the terminal and are specific to Nutch. Nutch prompts detailed instructions on how to use the Nutch commands if they are input incorrectly. Commands can be run recursively with the scheduling tool Cron. Crawls can be scheduled monthly to ensure that the records remain up to date. This can also be applied to the download script in Python. Clustering is a key advantage of Nutch as crawls can be run from multiple machines simultaneously. Remote servers are recommended for recursive crawls to provide adequate RAM and storage for the database of records.

4.3 Challenges & Limitations

Scalability and modularity are great from a theoretical and health domain perspective as Nutch can be applied to various different contexts however, this increases the usage difficulty. There are default settings provided by Nutch but there is little guidance on how to run a crawl for a specific context, it is completely up to the user how these preferences should be set. The task of acquiring specific documents is not a usual application of a web crawler, particularly Nutch. One of it's key strengths is the ranking characteristic however this is not necessary when the search criteria is so specific. A smaller scale web crawler may have been more appropriate for this task but in it's ease of use, it would sacrifice scalability.

This paper aims to provide a detailed configuration, in conjunction with education on the Nutch basics as the concepts can be a difficult to grasp. There is vast amounts of documentation available for Nutch but it is often not presented in context with applications. There are potentially other properties in Nutch that are useful to a document retrieval crawl but due to the range of Nutch properties it is challenging to ensure that the appropriate ones are utilised in the task at hand.

The main priority of automating a process is make a manual process more efficient. As these crawls were run locally with varying internet connectivity and limited storage, it is difficult to asses how efficient the crawl process is. This is a key limitation of web crawling as configuration can be time consuming and the crawl runs may also be computationally expensive. Despite this, with more adaptions and refinement an automated crawler is a much more efficient method for document retrieval.

4.4 Future Research

When looking through the Nutch default properties, the capabilities are vast and can be used to a much greater capacity than used in this paper. To fully and efficiently apply Nutch to the case of health policy crawling, a more thorough examination of Nutch is recommended to ensure that efforts in applications are not redundant. Apache Nutch welcomes plugins made by users and there may be existing plugins designed to fit the task of downloading specific file types such as PDFs and storing them in this format. With More in-depth research, experimentation and involvement in the Nutch community, these capabilities would be realised. Future research into utilising the raw text data from the downloaded documents for NLP research is also recommended. Utilising the raw data provided by Nutch is much more efficient that separately downloading and storing whole PDF files.

4.5 Conclusion

Health Policy research requires efficient access to documents to ensure that organisational bodies remain consistent and reliable in their procedures. Inconsistencies in the drug approval process can be detrimental. A web crawler for automated document retrieval in the context of EMA is presented to speed up the process of downloading individual documents from EMA and to provide a framework which can be applied to improve the process across the domain. In the configuration of this crawler, regex filters are applied however, an understanding of the target website architecture is required to utilise these filters effectively. The availability of document archives and overviews are useful in a successful Nutch configuration for health document retrieval. Further exploration into the properties and plugins available in Nutch will lead to a more refined web crawler. Despite the challenges in configuration, it remains evident that the use of a web crawler is significantly superior to the manual collection of online documents for Health Policy research.

References

[1] R. van Dinter, C. Catal, and B. Tekinerdogan, “A decision support system for automating document retrieval and citation screening,” *Expert Systems with Applications*, vol. 182, p. 115261, 2021.

[2] W.-K. Hon, M. Patil, R. Shah, S. V. Thankachan, and J. S. Vitter, *Indexes for document retrieval with relevance*, pp. 351–362. Springer, 2013.

[3] D. Parry, “A fuzzy ontology for medical document retrieval,” in *ACM International Conference Proceeding Series*, vol. 54, pp. 121–126.

[4] E. M. Beller, J. K.-H. Chen, U. L.-H. Wang, and P. P. Glasziou, “Are systematic reviews up-to-date at the time of publication?,” *Systematic Reviews*, vol. 2, no. 1, p. 36, 2013.

[5] H. Wood, A. O’Connor, J. Sargeant, and J. Glanville, “Information retrieval for systematic reviews in food and feed topics: a narrative review,” *Research synthesis methods*, vol. 9, no. 4, pp. 527–539, 2018.

[6] C. Lynch, “The retrieval problem for health policy and public health: knowledge bases and search engines,” *Journal of Urban Health*, vol. 75, no. 4, pp. 794–806, 1998.

[7] T. C. Wild, B. Pauly, L. Belle-Isle, W. Cavalieri, R. Elliott, C. Strike, K. Tupper, A. Hathaway, C. Dell, D. MacPherson, C. Sinclair, K. Karekezi, B. Tan, and E. Hyshka, “Canadian harm reduction policies: A comparative content analysis of provincial and territorial documents, 2000–2015,” *International Journal of Drug Policy*, vol. 45, pp. 9–17, 2017.

[8] D. Demner-Fushman, M. D. Kohli, M. B. Rosenman, S. E. Shooshan, L. Rodriguez, S. Antani, G. R. Thoma, and C. J. McDonald, “Preparing a collection of radiology examinations for distribution and retrieval,” *Journal of the American Medical Informatics Association*, vol. 23, no. 2, pp. 304–310, 2015.

[9] S. J. Darmoni, S. Pereira, S. Sakji, T. Merabti, [U+FFFD] Prieur, M. Joubert, and B. Thirion, “Multiple terminologies in a health portal: automatic indexing and information retrieval,” in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 255–259, Springer.

[10] R. A. Vreman, J. C. Bouvy, L. T. Bloem, A. M. Hövels, A. K. Mantel-Teeuwisse, H. G. M. Leufkens, and W. G. Goettsch, “Weighing of evidence by health technology assessment bodies: Retrospective study of reimbursement recommendations for conditionally approved drugs,” *Clin Pharmacol Ther*, vol. 105, no. 3, pp. 684–691, 2019.

[11] R. A. Vreman, L. T. Bloem, S. van Oirschot, J. Hoekman, M. E. van der Elst, H. G. Leufkens, O. H. Klungel, W. G. Goettsch, and A. K. Mantel-Teeuwisse, “The role of regulator-imposed post-approval studies in health technology assessments for conditionally approved drugs,” *Int J Health Policy Manag*, 2020.

[12] R. A. Vreman, H. Naci, W. G. Goettsch, A. K. Mantel-Teeuwisse, S. G. Schneeweiss, H. G. M. Leufkens, and A. S. Kesselheim, "Decision making under uncertainty: Comparing regulatory and health technology assessment reviews of medicines in the united states and europe," *Clin Pharmacol Ther*, vol. 108, no. 2, pp. 350–357, 2020.

[13] I. J. Marshall, B. Nye, J. Kuiper, A. Noel-Storr, R. Marshall, R. Maclean, F. Soboczenski, A. Nenkova, J. Thomas, and B. C. Wallace, "Trialstreamer: A living, automatically updated database of clinical trial reports," *Journal of the American Medical Informatics Association*, vol. 27, no. 12, pp. 1903–1912, 2020.

[14] J. Kaur, M. Yusof, P. Boursier, and J.-M. Ogier, "Automated scientific document retrieval," in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 5, pp. 732–736, IEEE.

[15] D. Trieschnigg, P. Pezik, V. Lee, F. De Jong, W. Kraaij, and D. Rebholz-Schuhmann, "Mesh up: effective mesh text classification for improved document retrieval," *Bioinformatics*, vol. 25, no. 11, pp. 1412–1418, 2009.

[16] P. Srinivasan, J. Mitchell, O. Bodenreider, G. Pant, F. Menczer, and P. S. Acd, "Web crawling agents for retrieving biomedical information," 2002.

[17] M. Kumar, R. Bhatia, and D. Rattan, "A survey of web crawlers for information retrieval," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 6, p. e1218, 2017.

[18] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz, "Evaluating topic-driven web crawlers," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 241–249.

[19] "8 most popular java web crawling and scraping libraries." Available at https://mobilemonitoringsolutions.com/8-most-popular-java-web-crawling-scraping-libraries/, journal=Mobile Monitoring Solutions, year=2019, month=Feb.

[20] M. Yadav and N. Goyal, "Comparison of open source crawlers-a review," *International Journal of Scientific and Engineering Research*, vol. 6, no. 9, pp. 1544–1551, 2015.

[21] S. M. Mirtaheri, M. E. Dinçktürk, S. Hooshmand, G. V. Bochmann, G.-V. Jourdan, and I. V. Onut, "A brief history of web crawlers," *arXiv preprint arXiv:1405.0749*, 2014.

[22] N. Kowsalya, "An approach of web crawling and indexing of nutch," *International Journal of Scientific and Engineering Research*, vol. 5, no. 11, pp. 766–772, 2014.

[23] R. Khare, D. Cutting, K. Sitaker, and A. Rifkin, "Nutch: A flexible and scalable open-source web search engine," *Oregon State University*, vol. 1, pp. 32–32, 2004.

[24] “Apache nutch wiki,” Feb 2021. Available at https://cwiki.apache.org/confluence/display/NUTCH/NutchTutorial.

[25] “Apache nutch tutorial,” Dec 2020. Available at https://cwiki.apache.org/confluence/display/NUTCH/Home.

Appendix: Nutch Properties

Table 5: Nutch Default Properties

Property	Value	Description
ftp.content.limit	1048576	The length limit for downloaded content using the http/https protocols, in bytes. If this value is non-negative (>=0), content longer than it will be truncated; otherwise, no truncation at all. Do not confuse this setting with the file.content.limit setting
db.fetch.interval	7776000	The maximum number of seconds between re-fetches of a page (90 days). After this period every page in the db will be re-tried, no matter what is its status
fetcher.threads.fetch	10	The number of FetcherThreads the fetcher should use. This is also determines the maximum number of requests that are made at once (each FetcherThread handles one connection). The total number of threads running in distributed mode will be the number of fetcher threads * number of nodes as fetcher has one map task per node
indexer.max.content.length	-1	The maximum number of characters of a content that are indexed. Content beyond the limit is truncated. A value of -1 disables this check.
db.ignore.internal.links	false	If true, outlinks leading from a page to internal hosts or domain will be ignored. This is an effective way to limit the crawl to include only initially injected hosts or domains, without creating complex urlfilters. See 'db.ignore.external.links.mode'.