NCL-Crawl: A Large Scale Language-specific Web Crawling System

Hafiz Muhammad Shafiq, Muhammad Amir Mehmood
Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan
{hafiz.shafiq, amir.mehmood}@kics.edu.pk

Abstract

There exist many cases that require language-specific web crawling, e.g., text corpus building in Natural Language Processing (NLP) domain and regional language search engine content crawling. In NLP, linguistics use text corpus for statistical analysis, checking occurrences or validation of linguistic rules within a specific language territory. Similarly, regional search engines use focused crawling to serve better quality results to the users. In this study, we build a system "NCL-Crawl" for large scale language specific web crawling using Apache Nutch crawler. For this purpose, we have customized Apache Nutch and added Compact Language Detector 2 (CLD2) module for language identification at runtime. The system also provides an option to specify minimum language bytes to avoid garbage collection in configured language. For evaluation, we have chosen the Urdu language and crawled 25,723 documents from the given seed and got very good quality pages with better accuracy. Our work is an effort towards building large scale text corpus for the NLP community especially for the low resource languages. In addition, regional search engines can effectively use NCL-Crawl for language specific web crawling.

1. Introduction

With the passage of time as Internet users are increasing, many regional search engines have appeared in the search engine market, e.g., Baidu, Yandex etc. In China, Baidu has 76.69% market share [1] and in Russia, Yandex has 45.16% market share [2]. These types of regional search engines require to crawl the WWW for a specific language at large scale. This approach not only provides better quality web documents of given language, but also helps to use minimal resources in terms of storage, bandwidth, and time [3]. Further, language-specific crawl is also required to build text corpus for linguistic researches and NLP applications. The advancement in NLP and Information Retrieval (IR) domain, e.g., summarization, cross-language information retrieval, etc., requires to build corpus in single or multiple languages at large scale [4][5].

There exists many open-source solutions to crawl World Wide Web (WWW) from small to large scale, e.g., curl, Apache Nutch, Scrapy, Heritrix etc. [6]. But for language specific crawl, none of these provides a concrete solution. In most cases, a new job is executed to find language information of crawled documents, which is both time as well as storage consuming. Moreover, language threshold based crawling, e.g., crawling documents with more than 50% Urdu content, is even more complex than former case. Some crawlers provide language filters but in most cases, it is based on web server response header and hence, it requires customization to find language information from crawled content. For instance, Apache Nutch provides "language-identifier" plugin to find language information but it is also based on web server response header.

This work is an effort to build Apache Nutch with CLD2 Language Crawl (NCL-Crawl) - A system using Apache Nutch Crawler and Compact Language Detector (CLD2) for language specific web crawling. NCL-Crawl aims to filter web-documents based on the content size in bytes of a particular language. Apache Nutch is an open-source large scale web-crawler and is developed in Java language that can be extended very easily. It has two major development branches, i.e., 1.x and 2.x [7]. We have used latter one for our experimentation. For language detection, we have integrated CLD2 with Nutch that can detect a maximum of three languages in a single document with percentage information [8]. Further, we do our customization in the fetcher module of Nutch to remove irrelevant documents at run time which also minimizes time and storage resources. For this purpose, we also added many new configuration parameters to set the language label and minimum bytes threshold.

To test our work, we collect Urdu language seed of 50 URLs from different domains and run the crawler for 40 iterations. We configure Urdu minimum threshold to 256 bytes and disable out-links. For politeness, a maximum of 50 URLs are selected in each iteration from a single domain. Our main findings in this study, are given below:

- **Yield Rate Statistics:** NCL-Crawl runs for 40 iterations and crawls 25,723 documents. From total fetched documents, 24,172 documents have Urdu content more than configured threshold. Crawling rate varies from 50 documents to 1300 during the experimentation.

- **Accuracy Measurements:** Overall 93.99% of the fetched documents have Urdu content greater than the configured threshold. For each iteration, accuracy varies from 86% to 99%.

The rest of our paper is organized as follows: In Section 2, we discuss existing work for language specific web crawling. Section 3 presents our methodology for Nutch customization and experimentation. In Section 4, experimental results are presented. Finally, we conclude our work in Section 5.

2. Related Work

For language-specific crawl for text corpus building and regional search engines, researchers have suggested various solutions. For instance, [9] has proposed a heuristics-based approach for focused web crawler. This approach uses pattern-based recognition algorithm to match the topic of crawled text. It requires a lot of space to save fetched data in each iteration and later analysis for pattern recognition. In [10], the authors have used Dictionary and Breadth-First algorithm for focused crawling to build Javanese and Sundanese Corpus. Their study shows that these two algorithms deliver the highest performance as compared to others in a focused crawl.

Further, [11] has used Semantic Similarity Vector Space Model for focused crawler improvement. Their results show better performance of focused crawler as compared to the Breadth-First model and VSM model. Similarly, [12] and [13] have used topic-based approach for focused crawling. In former, the authors have built a classifier that evaluates the relevance of a given document with respect to the topics and in latter work, a weight table is constructed with topic frequency to check the similarly of a web page.

For language-specific crawl, [14] has used linguistic graph analysis approach for crawling. The authors have analyzed web data from large crawl with specific language attributes for selection strategies. Moreover, [15] has used language locality in selecting the crawl paths from a large space of Thai weblogs for specific web crawl. Their work achieve higher performance than a naive Breadth-First crawling strategy.

Apache Nutch is one of the most matured web crawlers and has been used extensively in the research area for web crawling [16]. In [17], the authors have optimized Apache Nutch for domain-specific crawling at large scale. During experimentation, they got a success rate of only 0.0015% due to sparse data distribution and duplicate content on the Web. In our approach, we have also used Apache Nutch to build language-specific web crawler.

3. Formatting instructions

In this section, we discuss our proposed approach for language-specific crawling. First, we briefly describe Apache Nutch crawler with different phases. Next, we discuss the existing challenges in Apache Nutch for language-specific crawling. After this, we describe our proposed approach for Nutch customization in this regard. Finally, we discuss our testing environment for customized Nutch.

3.1 Apache Nutch Crawler Overview

Apache Nutch is an open-source distributed crawler to crawl the web at large scale. There exist two major versions of Nutch namely 1.x and 2.x. The latter one differs from the former with the addition of Apache Gora as a storage abstraction layer that allows to use different NoSQL databases, e.g., Hbase, Cassandra, etc., [18]. We have used Apache Nutch 2.x branch in this study. Further, each cycle of Nutch consists of many phases to complete a job as shown in Figure 1. Each of these phases have been described below:

Inject & Generate: The *inject* phase is the first phase where selected seed URLs are provided and crawler starts crawling by introducing some default score to URLs. This step is very important because the crawler will grow and fetch new web-pages based on the initial seed. The next phase in Nutch is *generate* phase where top URLs are marked for fetching based on the assigned score to URLs. Note that this score is the default for the first iteration but later on, it is calculated in *updatedb* phase of Nutch for each next iteration.

Fetching: In this phase, the crawler requests the marked URLs (in the generator phase) and fetches HTML of these pages from the World Wide Web (WWW). This job is multi-threaded and one can control the number of threads via Nutch configuration. There exist many controls in Nutch for various purposes in this phase, e.g., age filter (*filter.age.timestamp*), size filter (http:content:limit), fetcher threads per host (*fetcher.threads.per.queue*) etc. At the end of this phase, complete downloaded HTML with headers is stored in configured storage back-end, e.g., Hbase etc.

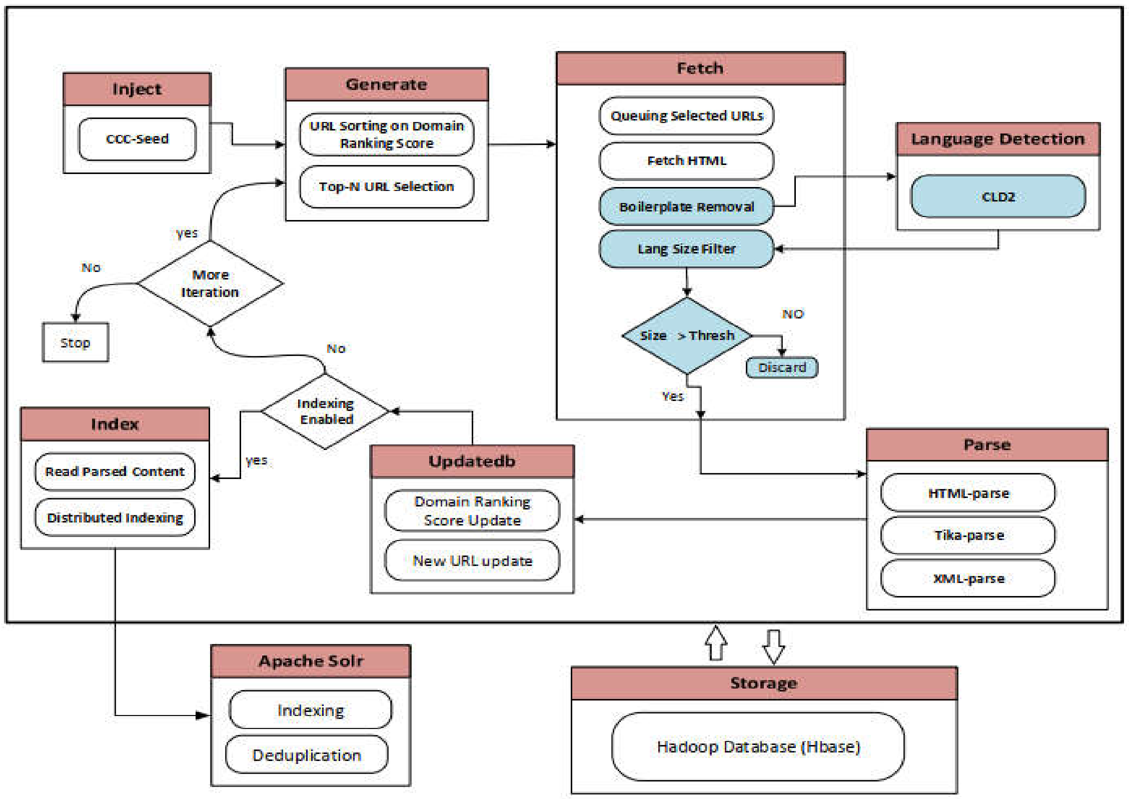
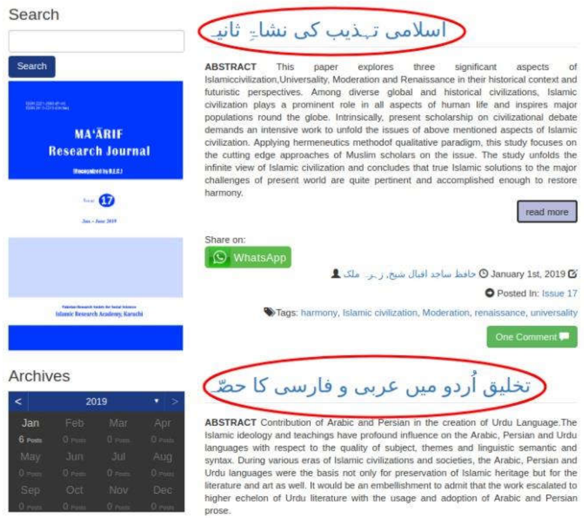


Figure 1: NCL-Crawl Execution Pipeline



اسلامی تہذیب کی نشاۃِ ثانیہ

ABSTRACT This paper explores three significant aspects of Islamiccivilization,Universality, Moderation and Renaissance in their historical context and futuristic perspectives. Among diverse global and historical civilizations, Islamic civilization plays a prominent role in all aspects of human life and inspires major populations round the globe. Intrinsically, present scholarship on civilizational debate demands an intensive work to unfold the issues of above mentioned aspects of Islamic civilization. Applying hermeneutics methodof qualitative paradigm, this study focuses on the cutting edge approaches of Muslim scholars on the issue. The study unfolds the infinite view of Islamic civilization and concludes that true Islamic solutions to the major challenges of present world are quite pertinent and accomplished enough to restore harmony.

حافظ ساجد اقبال شیخ, ز.ر. ملک January 1st, 2019

Posted In: Issue 17

Tags: harmony, Islamic civilization, Moderation, renaissance, universality

تخلیق اُردو میں عربی و فارسی کا حصّہ

ABSTRACT Contribution of Arabic and Persian in the creation of Urdu Language.The Islamic ideology and teachings have profound influence on the Arabic, Persian and Urdu languages with respect to the quality of subject, themes and linguistic semantic and syntax. During various eras of Islamic civilizations and societies, the Arabic, Persian and Urdu languages were the basis not only for preservation of Islamic heritage but for the literature and art as well. It would be an embellishment to admit that the work escalated to higher echelon of Urdu literature with the usage and adoption of Arabic and Persian prose.

Figure 2: A sample webpage with very small Urdu

Parsing: As discussed earlier, each crawled document consists of various levels of information, e.g., raw content, i.e., HTML source code, request/response headers, etc. The parse phase of Nutch parses each of this information of crawled data and saves them separately in the configured database. There are different parser plugins available in Nutch, e.g., html−parser, tika−parser, xml−parser that can be configured via Nutch configuration file.

UpdateDB & Indexing: After parsing, the next phase of crawling is to update the database using parsed documents. Many types of activities are performed in this phase, e.g., addition of in-links/outlinks, page score calculation, extra markers removal etc. There exist many configuration options for each of these actions to enable/disable these controls or to add some scoring plugin etc. For instance, db.ignore.external.links configuration parameter is used to allow the addition of external links, i.e., outlinks, in the database. Later on, these URLs get a mark possibility for fetch in *generate* phase based on their score. This is the last major phase of Nutch in the crawling cycle and the crawler can jump back to *generate* phase from here for the next cycle. Nutch also provides an indexing phase to index and search crawled content via some text search platform, e.g., Apache Solr or Elastic Search, etc. This phase should be executed after *updatedb* to include current crawled documents in index.

3.2. Language Specific Crawl and Challenges in Nutch

Although, Nutch provides language identification plugin ”language-identifier” to find language details of crawled documents, however, this plugin is based on a web-server response header and is not reliable. In most websites, this header is not properly set and in some cases, it is not even available. Further, Nutch also does not provide language information for multilingual pages with their percentage distribution.

To crawl specific websites only for text corpus building and regional search engine content, Nutch provides an option to disable out-links and crawl inlinks completely. For this purpose, one has to configure *db.ignore.external.links* property to true in Nutch configurations. Despite the fact that this approach will crawl all documents of given seed, but it cannot filter low quality documents w.r.t. given language, and hence, will cause garbage collection. For instance, Figure 2 shows such a sample page that has English as a primarily language and Urdu as a secondary language. Although, this document has Urdu content but it is of very small size, i.e., bytes. Such documents should be filtered for applications that requires very rich content in Urdu language. Unfortunately, Nutch does not provide any such option to avoid garbage collection for language specific crawling.

3.3. Nutch Customization for Language specific Crawl

In this section, we have discussed the customization of Apache Nutch for language specific crawling. First, we discuss the addition of language detection tool in Nutch and later, we discuss about the implementation of minimum language size filter to avoid garbage collection.

3.3.1. Language Detection Tool. For language specific crawling, the first and most important step is tool selection for language identification of a given language. For this purpose, there exist many open-source tools e.g., *langid, langdetect, ldig* and CLD2 [19][20][8]. Each of these tools has its own limitations and requirements. In our case, we have selected CLD2 for language identification of crawled content. CLD2 accepts only UTF-8 encoded strings and can identify 161 different languages. For a given text, it can detect upto maximum of three languages along with their percentage and total bytes. The percentage information can help to apply a minimum size filter for configured language as discussed later.

To add CLD2 module in Nutch, we decided to detect document language at runtime, and if a document is irrelevant, we truncate it at the spot. This strategy not only helps to remove documents that are not in the required language but also helps to save storage. For this purpose, we have customized Apache Nutch fetcher module that actually crawls the documents from WWW. In this module, fetched content is parsed via *Boilerpipe* library to get the main article of document. *Boilerpipe* is an open-source library developed for boilerplate removal from HTML documents [21]. Later, this extracted text is sent to CLD2 module that returns language information of the document.

3.3.2. Minimum Size Filter. In order to implement minimum size filter to avoid garbage collection, as already discussed, we cannot directly use CLD2 percentage value as a minimum threshold without language bytes information. For example, if there are two documents with a content of 1 MB and 1KB respectively and CLD2 returns 10% Urdu in both cases, then in first case, Urdu bytes are 100 KB while in second, these are just 100 bytes. Thus, first document is more rich with Urdu as compared to the second one. To cater this problem, we find language bytes from the CLD2 output using following equation:

$$LangBytes = \frac{(total\ bytes) * (lang.percentage)}{100}$$

Bytes information for above discussed documents will be as follows w.r.t. this equation:

$$Doc1\ LangBytes = \frac{(1{,}024{,}000)\ *\ (10)}{100} = 100KB$$

$$Doc2\ LangBytes = \frac{(1{,}024)\ *\ (10)}{100} = 100B$$

In order to configure language and minimum language threshold, we introduce few new configuration parameters, e.g., *filter.lang.label* and *filter.lang.minSize.bytes* etc. Complete details of all new configuration parameters are given in second part of Table 1 with default values and description. Lastly, Figure 1 shows complete workflow of Apache Nutch with language filter and minimum size filter. Our main contributions are highlighted with light blue color in the diagram.

3.4. Testing Environment

To test our customized Nutch crawler, we set up a small size Hadoop/Hbase cluster with 3 worker nodes and run it for 5, 10, 20, 30 and 40 iterations. We select Urdu as a test case language, and for seed, we collect 50 number of URLs from different Urdu domains. To avoid garbage collection, minimum threshold for language

Table 1: Configuration changes for testing environment with new language specific parameters

Type	Property	Value Options	Test value	Description
Default	db.ignore.internal.links	true, false	false	Enable/ disable internal links
Default	db.ignore.external.links	true, false	true	Enable/ disable external links
Default	generate.max.count	numeric	50	Maximum links from single domain in each iteration
New	filter.lang.enable	true, false	true	Enable/disable language filter
New	filter.lang.label	language label	Urdu	Language name to be filtered
New	filter.lang.minSize.enable	true, false	true	Enable/ disable minimum size filter
New	filter.lang.minSize.bytes	numeric (bytes)	1	Minimum language bytes, i.e., threshold
New	filter.lang.maxSize.enable	true, false	false	Enable/ disable maximum size filter
New	filter.lang.maxSize.bytes	numeric (bytes)	-	Maximum language bytes
New	filter.lang.minPercentage.enable	true, false	false	Enable/ disable language percentage filter
New	filter.lang.minPercentage.limit	Numeric (%)	-	Minimum language percentage

size filter is set to 256 bytes. In each iteration, the crawler selects a maximum 2,500 URLs (topN) from all domains, and from single domain, a maximum of 50 URLs are marked for politeness via *generate.max.count*. In addition, instead of manually checking crawl documents for accuracy measurement, we index all crawled documents in Apache Solr that is an open source full text search engine [22]. Relevant documents are retrieved using query filter present in Apache Solr. Important configuration parameters with their test-case values are given in Table 1.

4. Results

In this section, we present experimental results from language-specific crawler. First, we discuss yield rate statistics and later, we discuss the accuracy measures of our proposed crawler.

4.1. Crawler Yield Rate

Yield rate statistics help to know the crawling rate at different intervals. After the crawling job completion, a total of 25,723 documents are successfully fetched, out of which, 24,174 documents have content in Urdu language with a percentage more than threshold. Figure 3 presents yield plot in our experimentation of customized crawler for overall successfully crawled documents and Urdu language documents vs the number of iterations. In each iteration, the number of crawled documents is very close to the total number of documents fetched in that iteration. It shows better accuracy of crawler in context of Urdu documents. The crawling rate varies from 50 to 1300 in this analysis. The

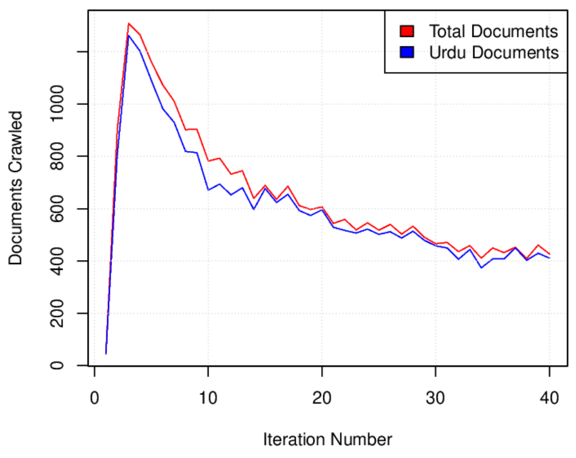


Figure 3: Crawler Yield Rate

decaying behavior in the figure shows that over time, available URL space, i.e., the list, is reducing as more and more URLs are crawled with time. It is due to the reason that we have disabled out-links as already discussed in Section 3.

4.2. Accuracy Measurements

In order to measure effectiveness of NCL-Crawl system, we use the percentage of relevant pages from total downloaded pages in each iteration as the accuracy

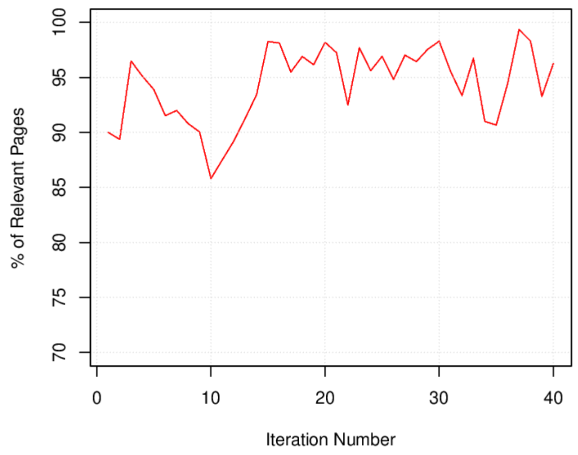


Figure 4: Accuracy Measure of Proposed Language

measurement. The relevant pages are those pages where Urdu language is found and where the percentage of Urdu content is greater than the configured threshold. Figure 4 shows the customized crawler accuracy measurement for each iteration, and it varies from 86% to 99% during the experimentation. Overall, the crawler accuracy is 93.99% which is a very good score.

In general, the system accuracy depends upon the seed collection and language threshold parameter. If the seed is refined to the given language and threshold is not very large, then the accuracy will be very high as observed in our current experimentation, and if seed is not very well refined in context of the given language and the threshold is set very high, then the accuracy will increase or decrease immoderately. As already discussed, this is a new feature added in the default Nutch crawler, hence we cannot compare our results with some existing feature in Nutch default version.

5. Conclusion

In this work, we endeavor to build a language specific web crawling system, i.e., NCL-Crawl, to assist the NLP community for textual corpus building and regional language web crawling at a large scale. For this purpose, we have customized the Apache Nutch fetcher class and added CLD2 language detection module to identify the language of crawled content at run time. For experimentation and evaluation of our work, we collect 50 seed URLs in Urdu language from different domains and run the crawler for 40 iterations. To avoid garbage collection, we set minimum size threshold to 256 bytes of Urdu. Total crawled documents are 25,174 that include 24,174 documents with Urdu language content more than threshold. The crawling rate varies from 50 to 1300 during the job execution. Overall accuracy is 93.99% and varies from 86% to 99%. In general, this accuracy is dependent on the given seed URLs and language threshold parameter and will vary with these two parameters. Lastly, this solution can be used for any language and threshold value; one has to just change configuration parameters only. In future, we plan to open source this system for research community.

6. Acknowledgement

This research work was funded by Higher Education Commission (HEC) Pakistan and Ministry of Planning Development and Reforms under National Center in Big Data and Cloud Computing.

7. References

[1] Search engine market share china — statcounter global stats. https://gs.statcounter.com/search-engine-marketshare/all/china/monthly-201808-201908, 2019.

[2] Search engine market share russian-federation — statcounter global stats. https://gs.statcounter.com/searchengine-market-share/all/russian-federation/monthly-201808-201908, 2019.

[3] Focused web crawler - wikipedia. https://en.wikipedia.org/wiki/Focused crawler, 2019.

[4] Natural Language Processing - Wikipedia. https://en.wikipedia.org/wiki/Natural language processing, 2019.

[5] Text Corpus - Wikipedia. https://en.wikipedia.org/wiki/Text corpus, 2019.

[6] Top 50 open source web crawlers for web mining. https://bigdata-madesimple.com/top-50-open-sourceweb-crawlers-for-data-mining/, 2019.

[7] WIKI Apache Nutch Web Crawler. https://wiki.apache.org/nutch/NutchTutorial, 2019.

[8] Compact Language Detector 2. CLD2owners/cld2. https://github.com/CLD2Owners/cld2, 2019.

[9] Joy Dewanjee. Heuristic approach for designing a focused web crawler using cuckoo search. International Journal of Computer Science and Engineering, 4(09):59–63, 2016.

[10] William Eka Putra and Saiful Akbar. Focused crawling using dictionary algorithm with breadth first and by page length methods for javanese and sundanese corpus construction. International Journal of Procedia Technology, 11:870–876, 2013.

[11] Yajun Du, Wenjun Liu, Xianjing Lv, and Guoli Peng. An improved focused crawler based on semantic similarity vector space model. Applied Soft Computing, 36:392–407, 2015.

[12] Ayar Pranav and Sandip Chauhan. Efficient focused web crawling approach for search engine. International Journal of Computer Science and Mobile Computing, 4(5), 2015.

[13] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. Focused crawling: a new approach to topic specific web resource discovery. International Journal of Computer networks, 31(11-16):1623–1640, 1999.
[14] Takayuki Tamura, Kulwadee Somboonviwat, and Masaru Kitsuregawa. A method for language-specific web crawling and its evaluation. International Journal of Systems and Computers in Japan, 38(2):10–20, 2007.
[15] Kulwadee Somboonviwat, Masaru Kitsuregawa, and Takayuki Tamura. Simulation study of language specific web crawling. In Proceedings of International Conference on Data Engineering Workshops (ICDEW'05), pages 1254–1254. IEEE, 2005.
[16] Apache nutch web crawler. http://https://nutch.apache.org/, 2019.
[17] Luis A Lopez, Ruth Duerr, and Siri Jodha Singh Khalsa. Optimizing apache nutch for domain specific crawling at large scale. In Proceedings of International Conference on Big Data (Big Data), pages 1967–1971. IEEE, 2015.
[18] Digitalpebble's blog: Nutch fight! 1.7 vs 2.2.1. https://gs.statcounter.com/search-engine-market-share, 2019.
[19] saffsd. Python's standalone language identification tool. https://github.com/saffsd/langid.py, 2017.
[20] Michal Danilak. langdetect: language-detection library to python. https://github.com/Mimino666/langdetect, 2017.
[21] Boilerpipe. https://code.google.com/archive/p/boilerpipe, 2019.
[22] Apache solr. https://lucene.apache.org/solr/, 2019.