CS6322.501- Information Retrieval
News Search Engine

Fahad Shaon (fxs121530)

Samira Farhin (sxf112330)

Mohammad Ridwanur Rahman (mxr127030)

Introduction

In this project we implemented a fully working news search engine that has crawling, page ranking, indexing and a easy to use simple user interface to retrieve the indexed document.

Our design motivation is to build modular system, where each module will work independently. We tried to design such that failure of one single module won't bring down the whole system.

Architecture

- Crawling Engine
    - NSECrawler
    - RSS Feed Reader
- Rank Analyzer
- Indexer
- Retrieval Engine and User Interface

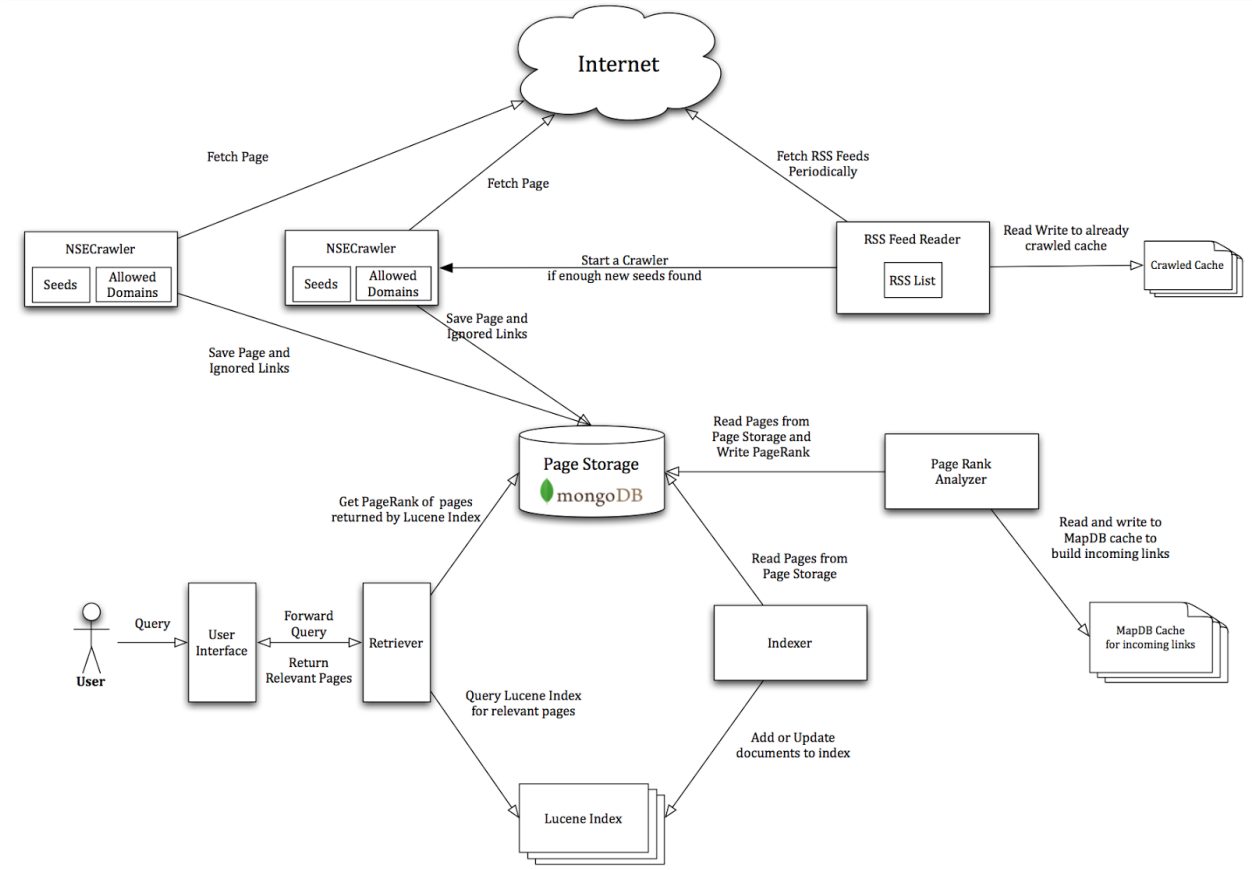

Figure: Overview of all the components

Crawling Component

NSECrawler

Basic component of crawling architecture is NSECrawler which is basically a crawler with fixed seeds and relevant parameters and tasked to crawl pages and saves the fetched pages. For simplicity we restricted our crawling to fixed domains and we also save the links to pages we didn't visited which makes up the frontier.

Input

List of seeds	Seed for crawling to begin with
Max Depth	the depth how much the crawler will crawl
List of allowed domains	Crawler will only crawl pages from these allowed domains.

Output

Pages	Pages that successfully crawled. Refer to "**Appendix A: Page Schema"** for details of what information are saved for a page
Ignored Links	Links that are ignored because of domain restriction. These list is a potential seed list for another crawler. Formally named as *frontier*. Refer to "**Appendix B: Ignored Links Schema"** for details of what information are saved about these links.

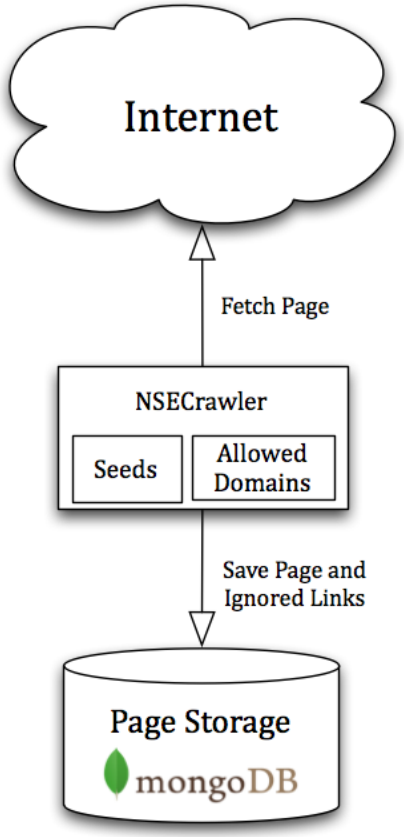

Figure: NES Crawler

RSS Feed Reader

On the course of building a news search engine we observed that every well known news provider make their recent news available in the form of RSS (**Rich Site Summary**, sometimes also called **Really Simple Syndication**). This sub-component leverages RSS to get the most recent news from the web.

Architecture:

RSS Feed Reader is continuously running process. It has a cache of already crawled links. In each iteration it fetches the provided rss feeds and parses for news links. If it find new news links then it starts new NSECrawler with new links as seed and max depth set to 1. In short it does a superficial crawling.

Input

RSS Feed List	List of RSS Feeds that need to crawled

Output

Crawling Job	Starts crawling job with newly discovered news.

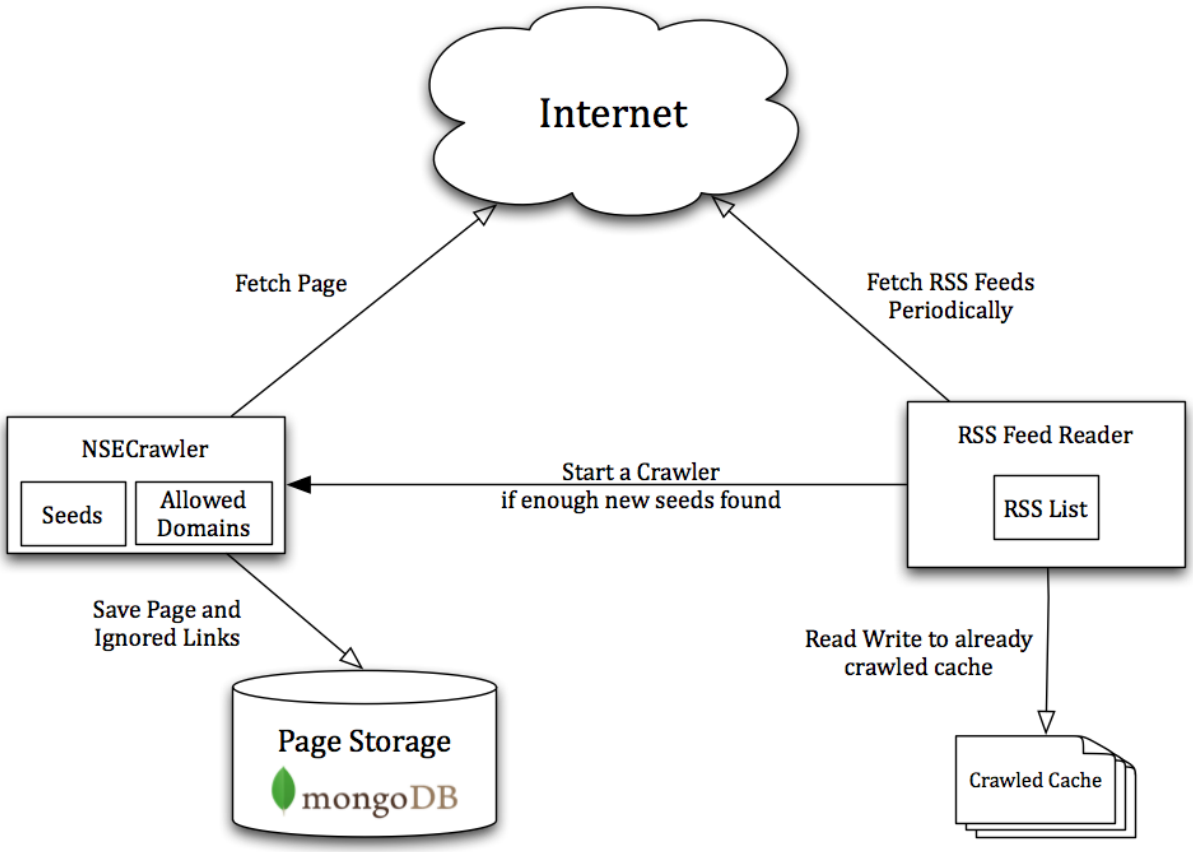

Figure: RSS Feed Reader

Rank Analyzer

As of writing this document we have crawled about 500,000 pages. And running PageRank on this enormous graph is itself a monumental challenge. We tried current state of the art *non-distribued* graph libraries like **GraphStream** (http://graphstream-project.org/) and **JUNG** the **Java Universal Network/Graph Framework** (http://jung.sourceforge.net/site/index.html) to compute PageRank of our already collected pages. These experiments failed with *java.lang.OutOfMemoryError*. No surprise though.

Next we investigate possibility of computing large graph in single computer. That leads us to the GraphChi (http://graphlab.org/graphchi/) project. We were successfully able to configure the library and run PageRank for smaller graphs. However, we couldn't make it run for our graph.

And finally we solved our problem by using disk based hash map, MapDB (http://www.mapdb.org/).

To successfully compute PageRank we need to know few property of each page of the document collection -

1. Outgoing link count
2. Pages that have link(s) pointed to a particular page.

However when we crawled the web we only know the outgoing links. So we build a URL to PageID tuples list, which can be efficiently queried by either PageID or URL.

PageID	URL
1	http://www.cnn.com/..
2	http://www.bbc.com/..
…	…

Table: PageID, URL tuple list.

Next we build TargetPage and Source Page, tuples list which is ordered By TargetPage.

Target Page	Source Page
2	1
3	1
4	3
..	..

Table: Target Page, Source Page tuple list.

Now we can find incoming links for all the pages. Next compute the PageRank as usual. And we normalize our PageRank 0.001 to 1.

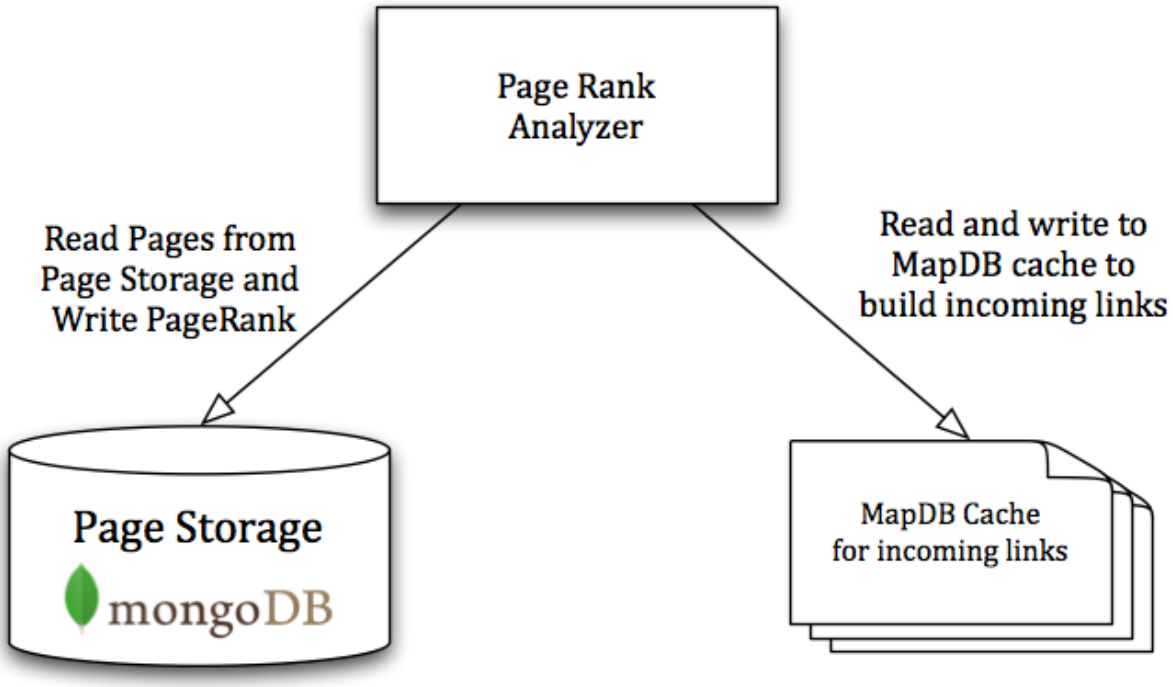

Figure: PageRank Analyzer

Indexer

This project utilizes Lucene for indexing the crawled pages. This component is also a continuously running process. It looks for any not yet indexed pages. If found then parse and add to index.

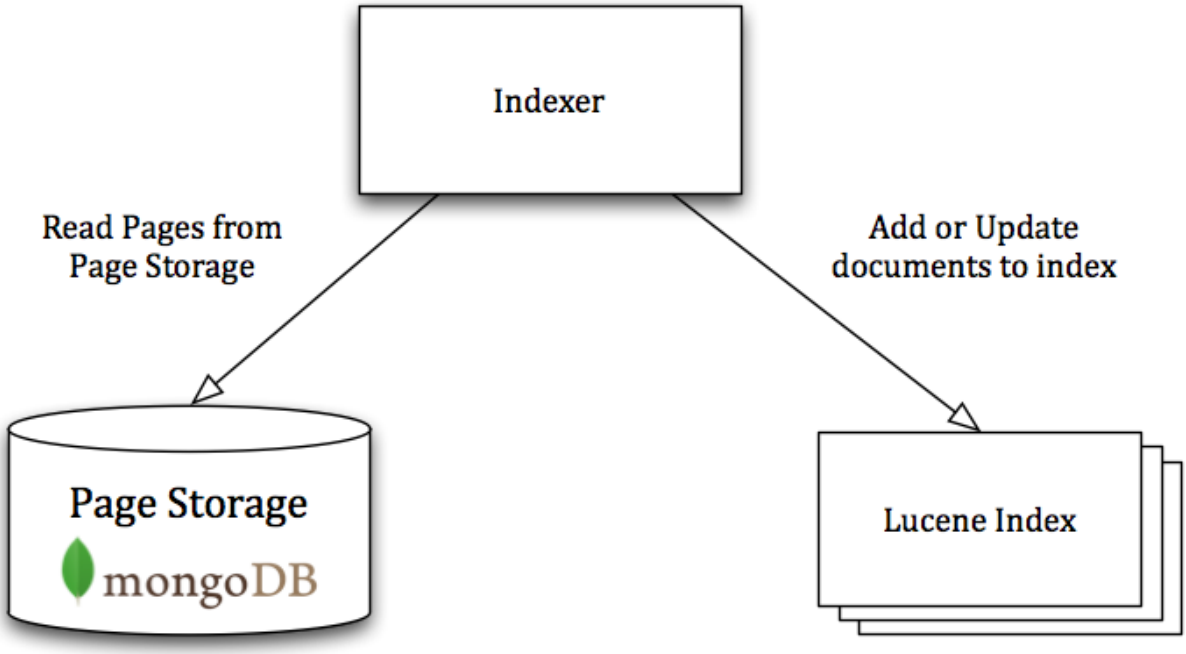

Figure: Indexer

Retrieval Engine and User Interface

Architecture:

Retrieval engine queries the Lucene indexes using Lucene API and retrieves relevant documents for a query. Then retrieval engine query the page database for page title, raw html and pre computed PageRank. Next it reordered few pages or results based on a computed score and send return it. This score is computed with

```
Score = Similarity score + PageRank
```

Where,

`Similarity Score` = Lucene provided hit score, that signifies how relevant this page is to query.

`PageRank` = PageRank of the page, this signifies how better linked a page is

Optionally Retrieval Engine also returns the original list of pages that was returned by Lucene.

Input

Query	The query of the user

Output

Ranked Pages	Relevant pages for user query based on score scheme
Lucene Ranked Page *(Optional)*	Lucene returned relevant pages for the query.

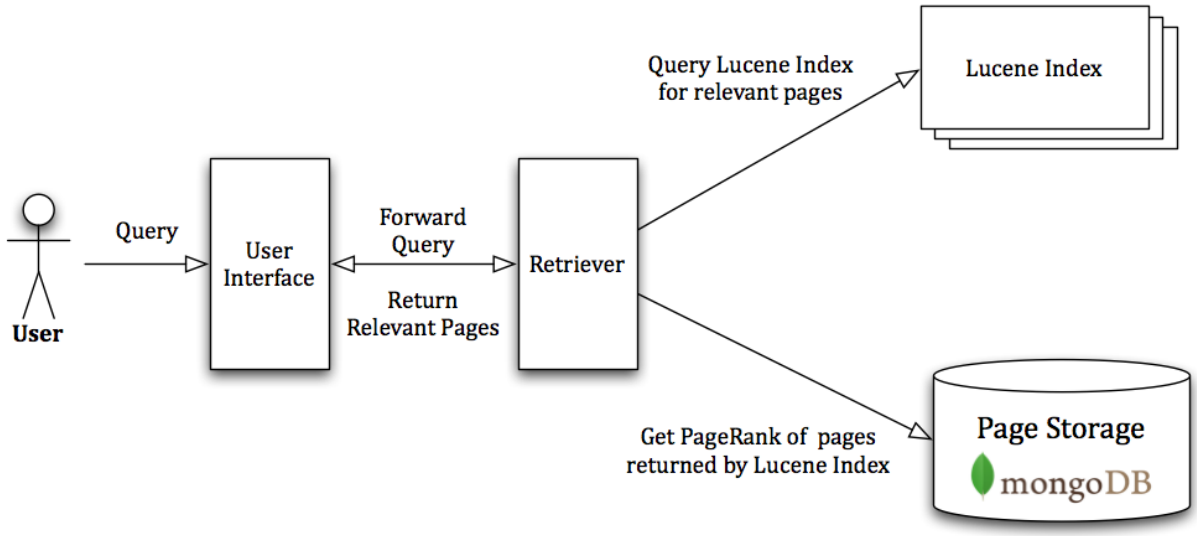

Figure: User Interface and Retriever

Statistics

Total pages crawled	574784
Total size of the document collection	79.229 GB
Average page size	142.89 KB

Page count per domain

www.huffingtonpost.com	327828
www.cnn.com	130958
abcnews.go.com	95973
www.foxnews.com	5577
www.afp.com	3178
www.cbsnews.com	2626
www.bbc.co.uk	2347
www.theguardian.com	1203
www.csmonitor.com	910
www.cbc.ca	729
news.yahoo.com	485
www.npr.org	447
www.spiegel.de	327
www.pbs.org	321
id.theguardian.com	240
www.rte.ie	184
www.usatoday.com	179
video.foxnews.com	166
www.euronews.com	137
hosted.ap.org	113
www.un.org	109
news.sky.com	104
m.csmonitor.com	88
edition.cnn.com	62

latino.foxnews.com	50
ssl.bbc.co.uk	49
www.bbc.com	43
feeds.theguardian.com	29
www1.spiegel.de	27
news.bbc.co.uk	22
feeds.bbci.co.uk	21
travel.cnn.com	21
rss.cnn.com	18
money.cnn.com	15
jobs.theguardian.com	13
www.nytimes.com	11
rss.csmonitor.com	11
inhealth.cnn.com	11
worldsport.blogs.cnn.com	10
help.npr.org	10
www.reuters.com	9

Open Source Libraries Used in the Project

Library	**Website**	**Used In Modules**
MapDB	http://www.mapdb.org/	PageRank Analyzer, RSS Feed Reader
MongoDB	http://www.mongodb.org/	All modules
Crawler4J	https://code.google.com/p/crawler4j/	Crawler
Rome	https://java.net/projects/rome/pages/Home	RSS Feed Reader

Lucene	http://lucene.apache.org	Indexer, Retriever
Jsoup	http://jsoup.org/	Indexer
Spark	http://www.sparkjava.com/	UserInterface

Appendix A: Page Schema

A saved page have title, html, url, crawled, indexed, pagerank, urlLinks.

Attribute	Description	Constraint
title	Title of the web page	
html	Raw html of the web page	
url	URL of the web page	Unique
crawled	Date when this page was crawled	
indexed	A flag indicating that the page is indexed or not	Indexed
pagerank	The PageRank value for this url	
urlLinks	List of links this page points to	

Example of a page document

```
{
    "title" : "Page Title ",
    "html" :  "Page Raw Html",
    "url" : "..page-url..",
    "crawled" : ISODate("2013-12-13T02:44:50.036Z"),
    "indexed" : 0,
    "pagerank": 0.123456789
    "urlList" : [
        {
            "url" : "out-link",
            "text" : "Link Text"
        },
        {
            "url" : "another-out-link",
            "text" : "Another Link Text"
        },
        ... [More Links] ...
    ]
}
```

Appendix B: Ignored Links Schema

A saved ignored links have title, html, url, crawled, indexed, pagerank, urlLinks.

url	URL that is ignored by the crawler
parent	URL of the page that links this page
added	Time when this information added

Example Ignore Link

```
{
      "url" : "url-of-ignored-page",
      "parent" : "url-of-page-that-point-to-this-page",
      "added" : ISODate("2013-12-13T02:44:49.847Z")
}
```