Documentation

Solution Overview

This project is divided into 5 major parts – Crawling, Indexing, get the most representative image, Searching and Clustering.

Crawling: crawls web pages recursively, starting at a seed URL that is specified by the user in the form “http://www.example.com”, user also has to be able to specify the crawling depth. Irrespective of the crawl depth a Jsoup Document is created for each URL and its contents extracted. For a url with a crawl depth of 1, contents of pages directly linked in the seed URL, are extracted and indexed along with the contents of the seed URL. The seed URL is considered to be of depth 0. This crawler implements the recursive crawling through depth first search.

Indexing: The contents of each document are tokenized using the TokenStream and the stop words are eliminated using the StandardAnalyzer. The output of the stop word elimination is stemmed using the PorterStemFilter(Porter Stemmer) and indexed.

Searching: User entered input search query is tokenized, stop word eliminated and stemmed. The stemmed string is returned to the search component which would compare and search in the indexed contents (folder "Index" in the context Path). If there is a match, the output returns the most relevant documents with their rank, URL, relevance score and Representative image.

Representative Image: All the Images of each relevant web page which were stored before is used to get a representative image for that particular web page.

Clustering:

K-Means: K- Means one of unsupervised learning technique to solve the clustering problem by classifying the given data set through a certain number of clusters by taking input from the user. In our program parameter K can be specified at the run time by the user. At the first run we select the random centroid and assign the given data set to one of its nearest clusters using Euclidean distance. And then update the new centroids by taking average of its data members in the clusters. We follow the same iteration till it reaches convergence i.e no data member in the cluster changes.

Software’s and Jar files used:

Apache Lucene: Lucene is an open source search library that provides standard functionality for analyzing, indexing and searching text-based documents.

Jars Files used in Lucene: lucene-analyzers-common-5.5.3, lucene-core-5.5.3, lucene-demo-5.5.3, lucene-queryparser-5.5.3.

Jsoup HTML parser: jsoup-1.10.1

Weka 3.8: For clustering

<u>commons-io-2.5:</u> File Manipulation utilities.

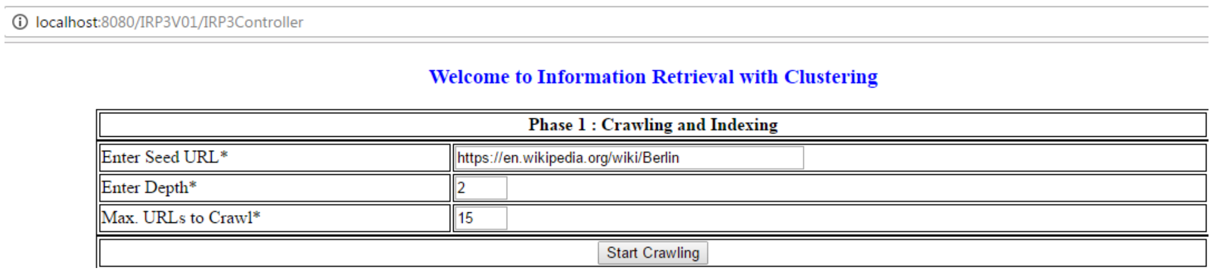
<u>Jstl-1.2</u>: JSP Standard Tag libraries.

Apache Tomcat server 7 or 8.

<u>**How can I use it**</u>

Deploy the war file in the appropriate server folder(User should have full file permissions on the directory where tomcat/app server is running).

1) Enter URL, Depth and No of URLs to Crawl. Click on "**Start Crawling**" button.

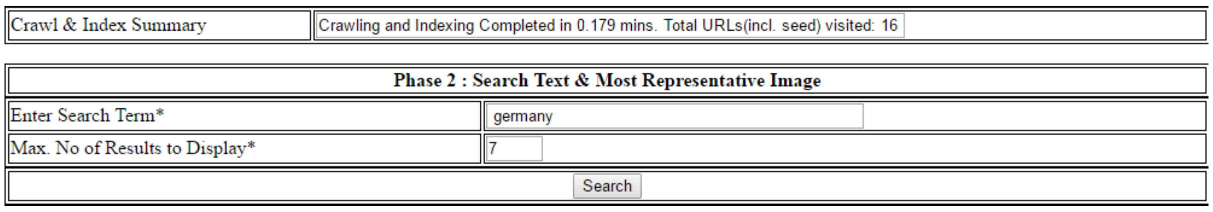


localhost:8080/IRP3V01/IRP3Controller

Welcome to Information Retrieval with Clustering

Phase 1 : Crawling and Indexing	
Enter Seed URL*	https://en.wikipedia.org/wiki/Berlin
Enter Depth*	2
Max. URLs to Crawl*	15
Start Crawling	

2) After Crawling completes, enter search query term and Max. Numbers of result to display in the field specified, and then click on "**Search"** button.



Crawl & Index Summary	Crawling and Indexing Completed in 0.179 mins. Total URLs(incl. seed) visited: 16

Phase 2 : Search Text & Most Representative Image	
Enter Search Term*	germany
Max. No of Results to Display*	7
Search	

3) Search result will be displayed with Rank, score, URL and most representatives Image of that URL.

Search Summary	Search Completed in 0.879 mins. Displaying :7 of Available Results: 16		
Rank	**Score**	**URL**	**Representative Image**
1	0.12151828	https://en.wikipedia.org/wiki/States_of_Germany	
2	0.08664481	https://en.wikipedia.org/wiki/Berlin_Wall	
3	0.07967218	https://en.wikipedia.org/wiki/Brandenburg_Gate	
4	0.0777334	https://en.wikipedia.org/wiki/Fernsehturm_Berlin	
5	0.06675003	https://en.wikipedia.org/wiki/Berlin#mw-head	
6	0.06675003	https://en.wikipedia.org/wiki/Berlin#p-search	
7	0.06675003	https://en.wikipedia.org/wiki/Berlin	

4) Clustering Part: Enter the number of clusters in the field specified and click on “**Cluster”** button. Result of clustering will be displayed with Cluster numbers, number of URLs in that cluster and respective Image for the URLs

Phase 3 : Clustering

● "Spherical" KMeans with Instance Normalization

Enter # of Clusters (A typical value = 3 for K-Means) *

Cluster

Clustering Summary	Clustering Completed in 0.074 mins. With 3 Clusters

Clusters	**URLs**	**No of URLs**	**Images per Cluster**
Cluster : 0	https://en.wikipedia.org/wiki/Berlin_Wall ~ https://en.wikipedia.org/wiki/Brandenburg_Gate ~	2	
Cluster : 1	https://en.wikipedia.org/wiki/States_of_Germany ~ https://en.wikipedia.org/wiki/Berlin#mw-head ~ https://en.wikipedia.org/wiki/Berlin#p-search ~ https://en.wikipedia.org/wiki/Berlin ~	4	
Cluster : 2	https://en.wikipedia.org/wiki/Fernsehturm_Berlin ~	1	

High Level Implementation

Crawler:

intialize: calls crawlUrl and closes the Indexwriter.

crawlUrl: A recursive function for crawling webpages based on crawlDepth and url. “pagesVisited” Contains a unique “set” of crawled URLs. Url normalization is also performed in this method.

“links” contains a “list” of URLs to be crawled for subsequent depths. It does not have duplicates. Hence normalized. Links with patterns like jar, zip, mp3 etc are excluded from adding to this list.

CrawlingUrl: Take out the text, title of particular file using jsoup. Send text for indexing.

Indexing:

porterStem: Porter Stem’s the text content retrieved by jsoup for each crawled url.

IndexWebPage: store the stemmed contents, url and title in a lucene document.

FileSearch:

searchInitializations: Initialise the variables.

SearchDoc : Searches max 10 relevant documents and gives the output. Also it calls representative image function to get the best representative image for the URL.

StemQuery: Input query is stemmed using porterstem.

GetImages: Download images and stores in server folder for each webpage.

RepImg:

ReprensatativeImage: Returns the most representative image of the URL.

CategoriseImg: Categorizes the image into various categories.

DocumentClustering

loadFromSearchedAndCreateFiles: We are iterating over Map which we got from search result. We are doing file operation to create text file of document (URL) content using TextDirectoryLoader class in weka. On this content we are getting Tf-Incidence matrix using StringToWordVector. This matrix will have class label generated since K- means is unsupervised we are deleting class labels using filters. This method produces the new data set, that we use it for cluster.

createClusters: Here we are normalizing that data using Normalize class in weka, on this data set we do cluster using SimpleKMeans class. Here we are passing K value which user enters.

Description of the Algorithm used in selecting Representative image of a particular web page

All the images of the resulting relevant web page(upon a search query entered and clicking on a search button) are stored in the working directory(context path). These images are input for representative image algorithm. Each image is given a score based on its attributes like height, width, ratio of height to width, type of the image.

Each image is categorized based on the keywords present in its title. Logo image(where title of the image contains “logo") advertisement(where title of the image contains “free, adserver, now, buy, join, click, affiliate, adv, hits, counter") icon(where title of the image contains "background, bg, spirit, templates") banner(where title of the image contains "banner, header, footer, button")

Images which do not belong to any of the above category is categorized as Representative image. If no image is assigned to Representative image category then the image from the next highest priority category which has the maximum score in that category is returned as Representative image.

Testing

This project has been tested on tomcat apache(6,7,8) and JDK versions(1.7 and 1.8).

Crawler:
TestCase1: Depth 0: Only the content of the seed url is indexed.
TestCase2: Depth 1: contents of pages in the seed URL that are directly linked are indexed and so on.
TestCase3: Normalization: Each url can be crawled and indexed only once.

Representative Image:
TestCase1: Representative image for a web page is displayed from the images of that web page.
TestCase2: Only one Representative image is selected for a web page.
TestCase3: Representative image is selected based on the criterion used in algorithm.

Clustering:
TestCase1: Cluster based representative image visualization of 10 or more relevant web pages.
TestCase2: Representative images to be extracted clustered urls.