Architecture of The Portuguese Web Archive Search System

Hugo Viana, Daniel Gomes and Miguel Costa

April 7, 2016

Arquivo.pt

Abstract

Arquivo.pt - The Portuguese Web Archive, is a free and public service that enables full-text search over more than 2 billion files archived since 1996.

In September 2013, the entire system of *Arquivo.pt* collapsed and information was irreparably lost. Besides, *Arquivo.pt* also had to renovate its team.

This situation caused several problems on the reconstruction and maintenance of the search system. During the recovery process of the service, a severe lack of technical documentation about the architecture of the search system was identified.

This report describes the software architecture of *Arquivo.pt* search system. The main difference from other web archiving systems is that it makes use of Lucene indexes to support full-text search and also URL search instead of the commonly used CDX files.

The outline of this report is divided in 4 chapters. First, it presents use cases that demonstrate how to use full-text, URL or advanced searches. Second, it describes the components and the interactions among them. Third, it presents the main operational procedures to maintain the system, such as how to deploy a collection and their indexes to production. Finally, it outlines which are the ranking features that are currently ordering the full-text search results.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Searching with Arquivo.pt

The Portuguese Web Archive project began in 2008 and it aims to preserve web contents of interest to the Portuguese community. It was based on the Internet Archive archive-access project tools (1), which are used by most web archives worldwide. However, we observed that these tools did not fulfill users requirements at several levels, from data integration to user interface design. Thus, we researched and developed a new web archive search engine (46).

This chapter illustrates the usage of *Arquivo.pt* through its several user interfaces.

Arquivo.pt entry page is illustrated on fig. 1.1.

Figure 1.1: Home page of *Arquivo.pt* (`http://arquivo.pt/index.jsp?l=en`).

The entry page contains a highlights area which has examples of important web content archived, and a video that provides an usage example about the service *Arquivo.pt*. Provides a bilingual interface (i.e. Portuguese or English).

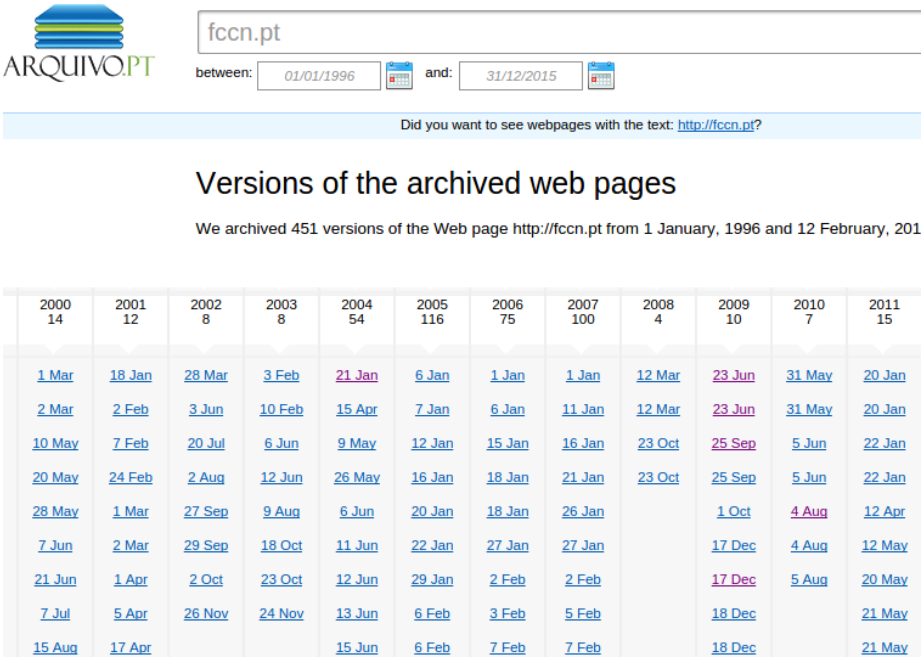
Arquivo.pt was designed to resemble a live-web search engine, in order to

produce a familiar search experience to users that are new to web archive services.

A live-web search engine generally supports full-text or URL searches. *Arquivo.pt* provides a similar experience but adding historical features.

Fig. 1.2 illustrates the historical search result page for the URL *fccn.pt*. The results page contains a table with all of the archived versions for the provided URL.

Figure 1.2: Url results page for the website "fccn.pt" (`http://arquivo.pt/search.jsp?l=pt&query=fccn.pt`).



A user can navigate across the web archived content and analysis their evolution through the years, as a time machine of web content.

Fig. 1.3 illustrates the full-text results page for the term *"fccn"*. The results list contains archived web content with the word *"fccn"*.

In both search pages, URL and full-text search, for delimiting the timespan of the search is provided two graphical datepickers. It is based on the Datepicker of jQuery UI library, but adapted to users of web archive services (45).

Fig. 1.4 illustrates the graphical datepicker developed for the web archive paradigm.

Figure 1.3: Full-text results page for the query "fccn" (`http://arquivo.pt/search.jsp?l=en&query=fccn`).

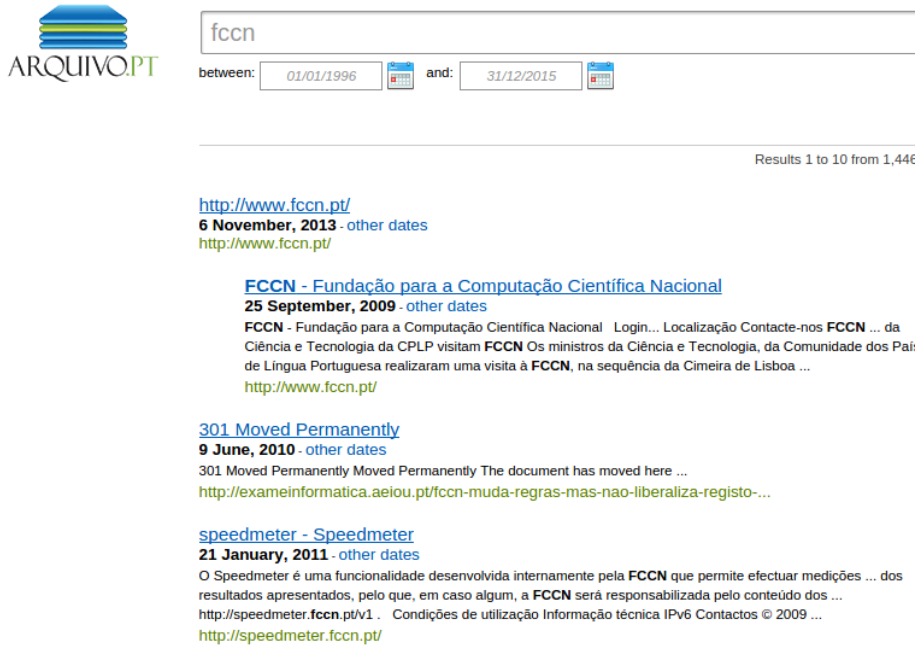


Figure 1.4: Datepicker adapted to web archive users.

Fig. 1.5 presents the advanced search page where it is provided to users several delimiters for improving their search results.

It contains the following fields:

Words

With these words: Looks up for web content which contains these words;

With these phrase: Looks up for web content which contains this phrase;

Without any of these phrase: Looks up for web content which not contains this phrase.

Date

Between: Looks up for web content by delimiting the date;

Sort by: Sets up the sort order.

Format

Show the pages in the format: Looks up for web content which are on a specific format (i.e. HTML, PDF etc..);

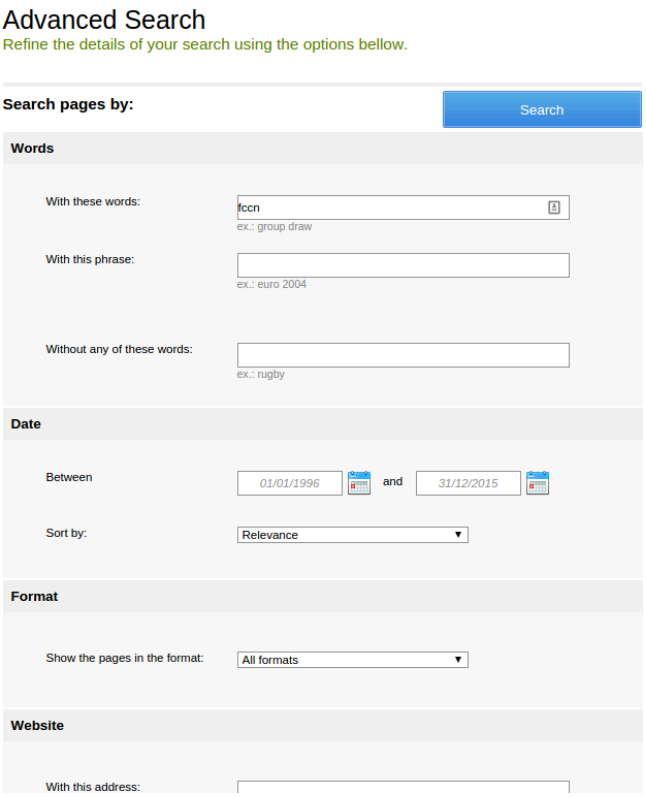
Website

With this address: Looks up for web content which contains this address;

Number of results

show: Sets up the number of results per page to be shown.

Figure 1.5: Advanced full-text search page (`http://arquivo.pt/advanced.jsp`).



Project and service information about *Arquivo.pt* is available at `http://sobre.arquivo.pt/portuguese-web-archive-2?set_language=en`.

Fig. 1.6 illustrates the informative home page of *Arquivo.pt*. It contains, for instance, publications related with the *Arquivo.pt*.

Figure 1.6: Informative site (`http://sobre.arquivo.pt/portuguese-web-archive-2?set_language=en`).

Chapter 2

Component-Based overview

2.1 System component overview

This section introduces a slight overview about the components of *Arquivo.pt*. It begins with an explanation of each component and then the workflow among them.

Arquivo.pt is divided in two platforms, which are:

Search system: is the full-text and URL search; (`http://arquivo.pt/?l=en`)

Info about the service: is the *Arquivo.pt* common management system (CMS). (`http://sobre.arquivo.pt/?set_language=en`)

2.1.1 Search system of *Arquivo.pt*

This subsection describes the components of the *Arquivo.pt* search system.

2.1.1.1 Broker

The Broker is "the manager" of the *Arquivo.pt* and it is responsible for delivering the ranked search results to users. The service contains redundancy between the Brokers, which means that two Brokers are listening and processing users requests.

2.1.1.2 QueryServer

QueryServer is the component that stores, looks up the indexes, ranks the results and then sends the results to the Broker. The indexes are distributed across multiple QueryServers.

2.1.1.3 DocumentServer

DocumentServer is the component that stores the harvest of web content in ARC format and make them accessible to the Broker. The harvest of web content are distributed across multiple DocumentServers.

2.1.1.4 LVS: Load Balancer

The load balancer balances the traffic distribution between the two Brokers. The service contains redundancy between the load balancer, which means two load balancer for processing users requests.

2.1.2 Info about *Arquivo.pt* service

This subsection describes the CMS component which manages all information about the service.

2.1.2.1 Info about the Arquivo.pt service

Arquivo.pt CMS is responsible for giving to know users the service. It contains several information about the service, such as documents produced, papers submitted and news.

2.1.2.2 LVS: fail over

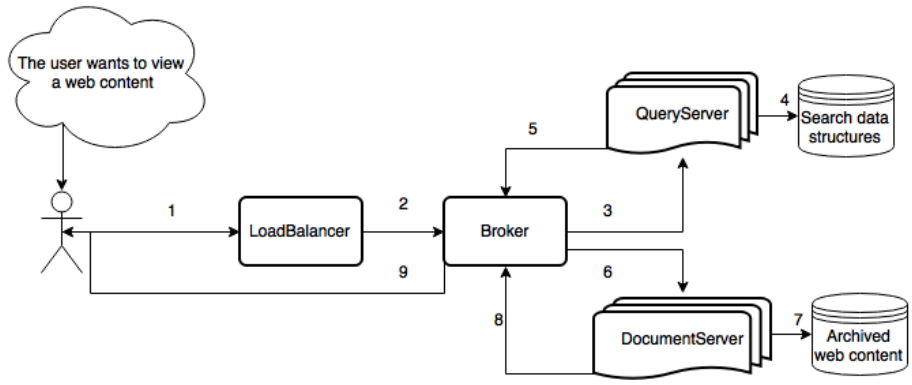
The fail over is a backup of the *Arquivo.pt* CMS in which the functions of the component are assumed by secondary system components when the primary component becomes unavailable.

2.2 A slight insight of the *Arquivo.pt* workflow

Based on the two platforms introduced in section 2.1, this section describes the workflow provided to users.

2.2.1 Searching for URL or terms

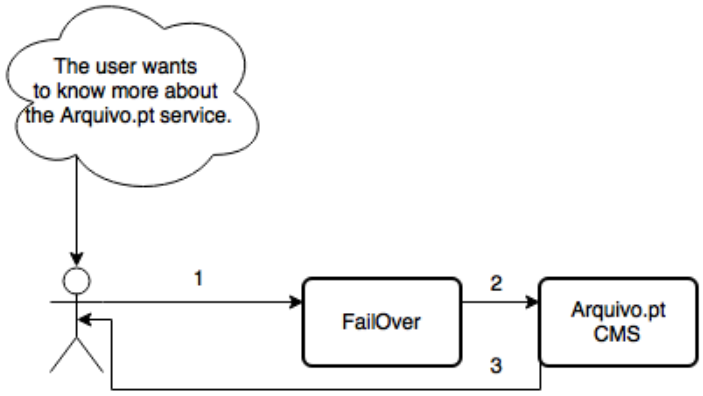
Figure 2.1: Workflow for searching by terms or URL.



1. The user submits a query to `http://arquivo.pt/?l=en`;
2. The load balancer picks one of the two Brokers to process the requests;
3. The Broker processes the query submitted and then broadcast it across QueryServers;
4. The QueryServer looks up for the query on the search data structures;
5. The QueryServer sends the ranked results list to the Broker;
6. The Broker processes the request and locates which DocumentServer stores the archived web content. Then, it requests the archived web content to the DocumentServer;
7. The DocumentServer obtains the archived web content;
8. The DocumentServer streams the archived web content to Broker;
9. The Broker reproduces the archived web content to user's browser.

2.2.2 Knowing more about the service

Figure 2.2: Workflow for knowing more about the service.



1. The user requests the page `http://sobre.arquivo.pt/?set_language=en`;
2. The failover checks if the primary components is available, if not the request is processed by the secondary;
3. The web site is sent to the user browser.

Chapter 3

Software Components and the Workflows

This chapter describes the bundle of software needed for setting up *Arquivo.pt* search system.

3.1 Software Components

The following describes the software components of *Arquivo.pt*.

nutchwax: manages the workflow of a URL search, processes full-text search requests and hosts the users interfaces;

wayback: manages the streaming web content sent from arcproxy, and processes a URL search request (Administrators Manual (50));

arcproxy: hosts a Berkeley database that maps the arcfile name into the server location (table 3.1 contains an example) and provides a stream connection for supplying the web content to the wayback;

spellchecker: is a mechanism that suggests alternatives queries (43);

browser provides navigation over archived web contents;

nutchwax-job-0.11.0-SNAPSHOT.jar: is a java standalone application for listening query requests from Broker and then returns a list with the 10k

most relevant results. It also provides the hadoop jobs for the indexing process;

Plone: *Arquivo.pt* content management system;

Zope: Web application server for running Plone CMS;

Tomcat: Web application server for running java web applications;

Piranha: Load-balanced generic service clustering environment;

Table 3.1: An ArcProxy database populated.

Name	**Server path**
FCCN-PT-HISTORICAL-ia400129.arc.gz	arquivo.pt:8080/browser/files/IAall/arcs/FCCN-PT-HISTORICAL-ia400129.arc.gz
FCCN-PT-HISTORICAL-ia400129.arc.gz	arquivo.pt:8080/browser/files/IAall/arcs//FCCN-PT-HISTORICAL-ia400129.arc.gz
FCCN-PT-HISTORICAL-ia400129.arc.gz	arquivo.pt:8080/browser/files/IAall/arcs/FCCN-PT-HISTORICAL-ia400129.arc.gz
FCCN-PT-HISTORICAL-ia400129.arc.gz	arquivo.pt:8080/browser/files/IAall/arcs/FCCN-PT-HISTORICAL-ia400129.arc.gz
FCCN-PT-HISTORICAL-ia400129.arc.gz	arquivo.pt:8080/browser/files/IAall/arcs/FCCN-PT-HISTORICAL-ia400129.arc.gz
FCCN-PT-HISTORICAL-ia400129.arc.gz	arquivo.pt:8080/browser/files/IAall/arcs/FCCN-PT-HISTORICAL-ia400129.arc.gz

3.2 Workflows

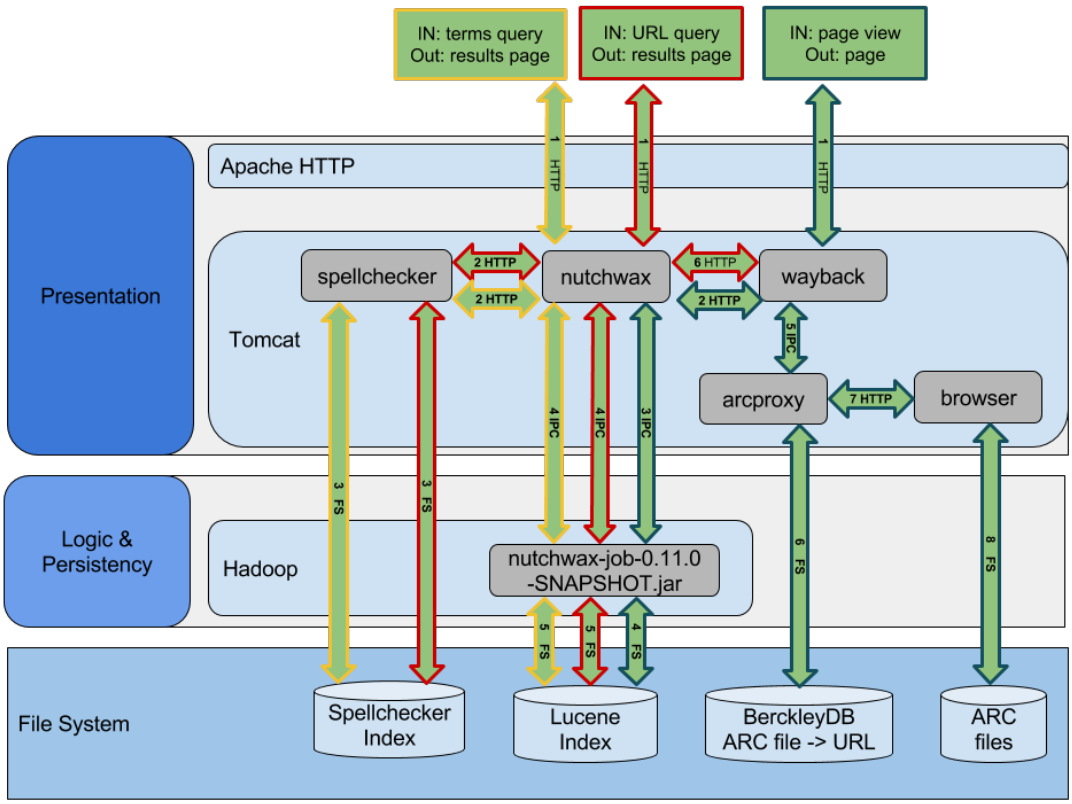
Fig. 3.1 details the software and the workflow of the *Arquivo.pt* service. The workflow of the software architecture is divided in:

Terms Query: Search by terms.

1. A user submit a terms query to Apache, which redirects it to nutchwax inside Tomcat.
2. The nutchwax asks the spellchecker for a query suggestion.
3. The spellchecker looks up the best suggestion and send it to the nutchwax.
4. The nutchwax broadcasts the request, which contains the terms query, across the nutchwax-job-0.11.0-SNAPSHOT.jar cluster.
5. Each nutchwax-job-0.11.0-SNAPSHOT.jar accesses the index and respond with a 10k ranked list of documents.

URL Query: Search by URL.

Figure 3.1: Software architecture of *Arquivo.pt*.



1. The user submit the URL query to Apache, which redirects it to wayback inside Tomcat.
2. The nutchwax asks the spellchecker for a query suggestion.
3. The spellchecker looks up the best suggestion and send it to the nutchwax
4. The nutchwax redirects the request to the respective nutchwax-job-0.11.0-SNAPSHOT.jar.
5. The nutchwax-job-0.11.0-SNAPSHOT.jar accesses the index and respond a list of URL versions.
6. The nutchwax redirects the list of URL versions to wayback.

Page View: Web content reply.

1. The user submit the URL of the page to view to Apache. Apache

redirects it to wayback inside Tomcat.

2. The wayback redirects the request to nutchwax.
3. The nutchwax redirects the request to the nutchwax-job-0.11.0-SNAPSHOT.jar.
4. The nutchwax-job-0.11.0-SNAPSHOT.jar accesses the index and respond a ARC filename and offset.
5. The wayback requests an ARC file to arcproxy.
6. The arcproxy accesses BerkeleyDB to get the URL of the ARC file.
7. The arcproxy gets the URL, which is a HTTP request to browser.
8. The browser returns the ARC file stored in the file system and then arcproxy starts streaming the web content to wayback. The wayback, in turns, stops the streaming when it gets the all of the requested web content.

Table 3.2 presents, for each web application, which are the communication protocols used for exchanging messages.

Table 3.2: Protocols used by *Arquivo.pt* applications.

Application name	**wayback**	**nutchwax**	**arcproxy**	**hadoopRpcServer**	**browser**
wayback		HTTP:8080	HTTP:8080		
nutchwax	HTTP:8080			HadoopRPC	
arcproxy	HTTP:8080				HTTP:8080
hadoopRpcServer		HadoopRPC			
browser			HTTP:8080		

Table 3.3 outlines how to use each Arquivo.pt servlet. For instance, to call the webapp **wayback** we have to produce the following:

wayback/19961013211156/http://www.telepac.pt/filosoft/teste.zip

Table 3.3: URL's to invoke directly each Tomcat web application hosted on Broker.

Webapp name	**URL example**
wayback/	19961013211156/http://www.telepac.pt/filosoft/teste.zip
nutchwax/	opensearch?query=exacturlexpand:http://www.sapo.pt/
arcproxy/	AWP-Roteiro-20090510220155-00000.arc.gz
spellchekcer/	checker?query=hola
browser/	files/FAWP10/FAWP-01-07-12-20120701150129893/arcs/IAH-20120701150204-00027-p12.arquivo.pt.arc.gz

Chapter 4

Arquivo.pt Software Implementation

Fig. 4.1 details the software and workflow of the *Arquivo.pt*.

4.1 System Components

This section details the software components of *Arquivo.pt*.

4.1.1 Broker

The Broker is "the manager" of the *Arquivo.pt*. The following web applications are hosted on the Broker:

nutchwax

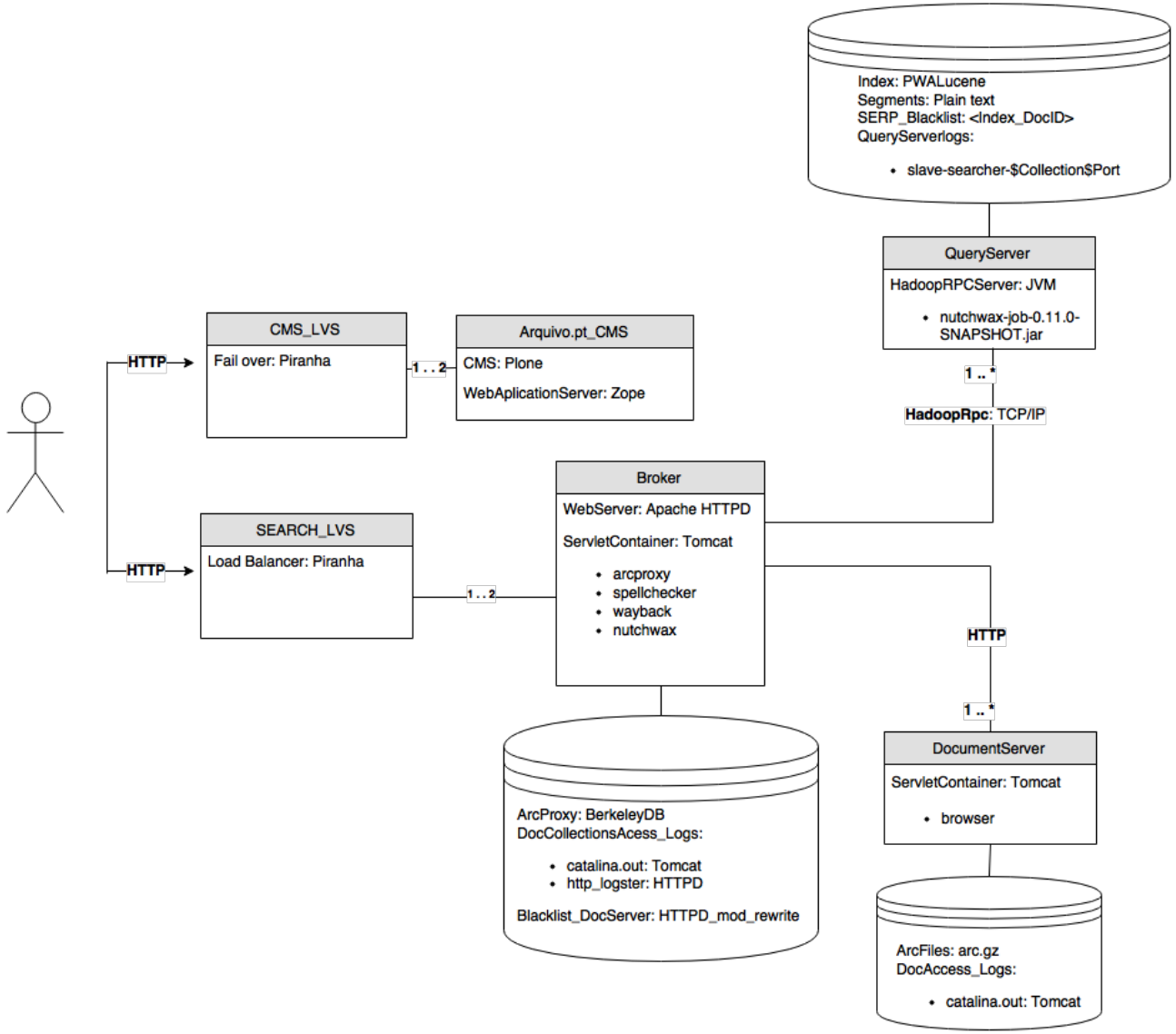
wayback

arcproxy

spellchecker

For further information about the web application, see chapter 3 that explains every web application used in *Arquivo.pt*.

Figure 4.1: The architecture of *Arquivo.pt*.



The Broker contains the following data structures:

ArcProxy:BerkeleyDB (more info in chapter 3)

catalina.out:Tomcat logs about query submission or components workflows are registered. For changing the log4j levels the following files are needed:

wayback /wayback/WEB-INF/classes/log4j.properties

nutchwax /nutchwax/WEB-INF/classes/log4j.properties

arcproxy /arcproxy/WEB-INF/classes/log4j.properties

spellchecker /spellchecker/WEB-INF/classes/log4j.properties

Table 4.1 contains examples of generated logs on catalina.out.

Table 4.1: Tomcat logs generated by Broker.

Feb 05, 2016 1:14:29 PM org.archive.wayback.webapp.AccessPoint logNotInArchive INFO: NotInArchive wayback 80 `http://gold.org/investment/statistics/gold_price_chart` 2016-02-05 13:14:30,283 INFO OpenSearchServlet - Index Information `http://arquivo.pt/wayback/id32984332index8` INFO: #session# 193.136.7.2 - - [05/Feb/2016:01:13:43 +0000] " GET /wayback/20100804063841/`http:/www.fccn.pt/` HTTP/1.1" 200 -1 "-" 2014-06-18 14:26:50,616 INFO Client - Retrying connect to server: p57.arquivo.pt/193.136.192.4:21116. Already tried 1 time(s). 2014-06-18 14:32:36,071 INFO PluginRepository - Plugins: looking in: /home/wayback/searcher/apache-tomcat-5.5.25/webapps/nutchwax/WEB-INF/classes/plugins

http_logster:HTTPD logs of HTTP request to *Arquivo.pt*. Table 4.2 presents several entries of http_logster file.

Table 4.2: HTTP_logster logs generate on Broker .

127.0.0.1	"GET /spellchecker/checker?query=teste&l=pt HTTP/1.1" 500 865 4384
127.0.0.1	"GET / HTTP/1.1" 200 3275 347499
127.0.0.1	"GET /img/highlights/figo.png HTTP/1.1" 200 1012 1936
127.0.0.1	"GET /img/search-icon.gif HTTP/1.1" 200 2922 8891
127.0.0.1	"GET /img/mec-web.png HTTP/1.1" 200 3309 1775
127.0.0.1	"GET /img/logo-pos.gif HTTP/1.1" 200 3077 1799
127.0.0.1	"GET /img/logo-fccn.png HTTP/1.1" 200 2218 6274
127.0.0.1	"GET /img/language-arrow.gif HTTP/1.1" 200 114 1696
127.0.0.1	"GET /img/search-reset.gif HTTP/1.1" 200 268 1996
127.0.0.1	"GET /img/search-inputtext.png HTTP/1.1" 200 211 2030
127.0.0.1	"GET /img/search-submit.gif HTTP/1.1" 200 1619 1776
127.0.0.1	"GET /img/box-background.gif HTTP/1.1" 200 4587 1802
127.0.0.1	"GET /img/box-mask.png HTTP/1.1" 200 358 2104
127.0.0.1	"GET /search.jsp?l=pt&query=porcaria&btnSubmit=Pesquisar+no+Arquivo HTTP/1.1" 200 5444 2766328

Blacklist_DocServer:HTTPD_mod_rewrite configuration file to manage accesses constraints of web content (apache HTTPD rewrite rules). Table 4.3 presents several rewrites rules.

Table 4.3: Rewrite rules to turn a web content inaccessible.

RewriteRule /wayback/id30257363index0 - [R=404,NC,L]
RewriteRule /wayback/id105349813index0 - [R=404,NC,L]
RewriteRule /wayback/id105560458index0 - [R=404,NC,L]

4.1.2 QueryServer

QueryServer is the component that stores and manages the indexes. The collection's indexes are distributed across multiple QueryServers.

QueryServer hosts the following java application.

nutchwax-job-0.11.0-SNAPSHOT.jar (more info in chapter 3)

The QueryServer contains the following data structures:

Index and Segments: stores the indexes and segments.

SERP_Blacklist: list of removed webpages from *Arquivo.pt*;

slave-searcher-$Collection$Port:HadoopRPCServer logs about users query.

Table 4.4 illustrates a few entries of generated logs on slave-searcher-$Collection$Port.

Table 4.4: Logs generated on file named *slave-searcher-10011.log*.

15/12/16 10:42:24 INFO ipc.Server: IPC Server handler 19 on 10016: starting
15/12/16 10:42:24 INFO ipc.Server: IPC Server handler 18 on 10016: starting
INFO searcher.LuceneQueryOptimizer:
15/12/16 10:45:01 Query:+(url:fccn anchor:fccn content:fccn title:fccn host:fccn)
+date:[0820454400 TO 1420070399]ĵ0.0
INFO searcher.LuceneQueryOptimizer:
16/02/05 16:07:10 Query:+exacturl:T4LL2V34WO5DK5HL2VHCAGHLK4ĵ0.1
+exacturl:LH5XGGMZBI2FO7OMTJ7BQE3UQAĵ0.1

4.1.3 DocumentServer

The DocumentServer is the component that stores the harvest of web content crawled in ARC format.

The following is the web applications hosted on the DocumentServer:

browser (more info in chapter 3)

It contains the following data structures:

ArcFiles:arc.gz stores the archived web content.

catalina.out:Tomcat logs about the accesses to the archived web content.

4.1.4 Arquivo.pt_CMS

Arquivo.pt_CMS is implemented using version 3.0.5 of Plone. Plone is a free and open source content management system (CMS) built on top of the Zope application server (20).

4.2 Workflow

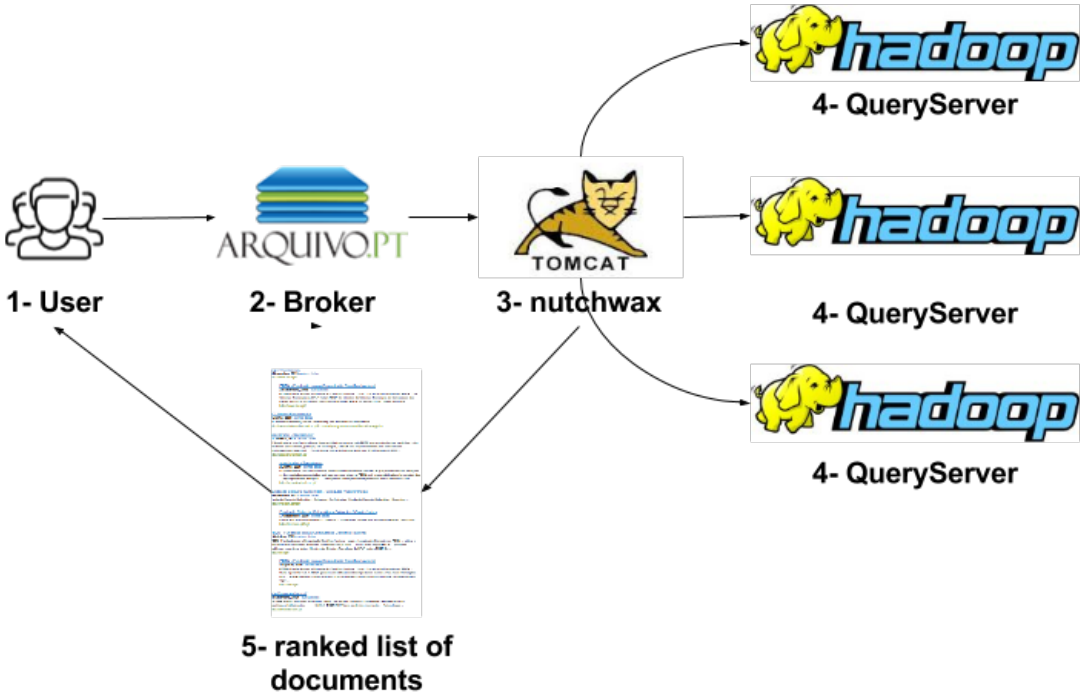
This section provides details about the communication among QueryServers, Brokers and DocumentServers, through use cases. *Arquivo.pt* provides the following workflows for users:

1. sends a query submission by URL.
2. sends a query submission by terms.
3. sends a query submission through advanced search.
4. requests a web content view.

4.2.1 Full-text query

Fig. 4.2 illustrates the workflow when a full-text search query is submitted to *Arquivo.pt*.

Figure 4.2: Workflow for a full-text search.



IN: "query: Luis Figo"

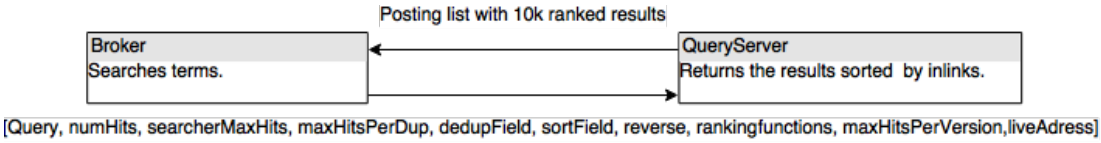
OUT: List of documents which contains "Luis Figo"

1. User submits a query request to Apache HTTP server.
   <http://arquivo.pt/search.jsp?query=%22Luis%20Figo%22>

2. Inside the Broker: Apache HTTP server redirects it to nutchwax.
   <http://arquivo.pt/nutchwax/opensearch?query="Luis%20Figo">

3. Inside the Broker: The nutchwax broadcast the processed request across QueryServers. Fig. 4.3 contains the protocol fields broadcasted.

Figure 4.3: Workflow between Broker and QueryServer



The communication is done over the Hadoop RPC protocol. The request parameters are described in the table 4.5.

Table 4.5: Parameters broadcasted across QueryServer (full-text search).

Parameter name	**Example**	**Description**
query	Luis%20Figo date:"19960101000000-20141231235959"	full-text search encoded with timestamp by default.
numHits	20	sets the number of listed results.
searcherMaxHits	10000	sets the max number of results for the posting list.
maxHitsPerDup	2	sets the max number of repeat results on the posting list.
dedupField	site	sets the maxHitsPerDup field filter.
sortField	null	sets the field name to sort (i.e sort by date ..).
reverse	false	sets the sort order by ascending or descending.
rankingfunctions	49=0.5939482, 37=0.34503713, 45=1.2592824, 34=0.023322608	sets the ranking features and its weights.
maxHitsPerVersion	1	selects max number of repeated version.
liveAddress	[p21.arquivo.pt/193.136.192.175:21136]	sets the hostname and ipaddress about the Broker.

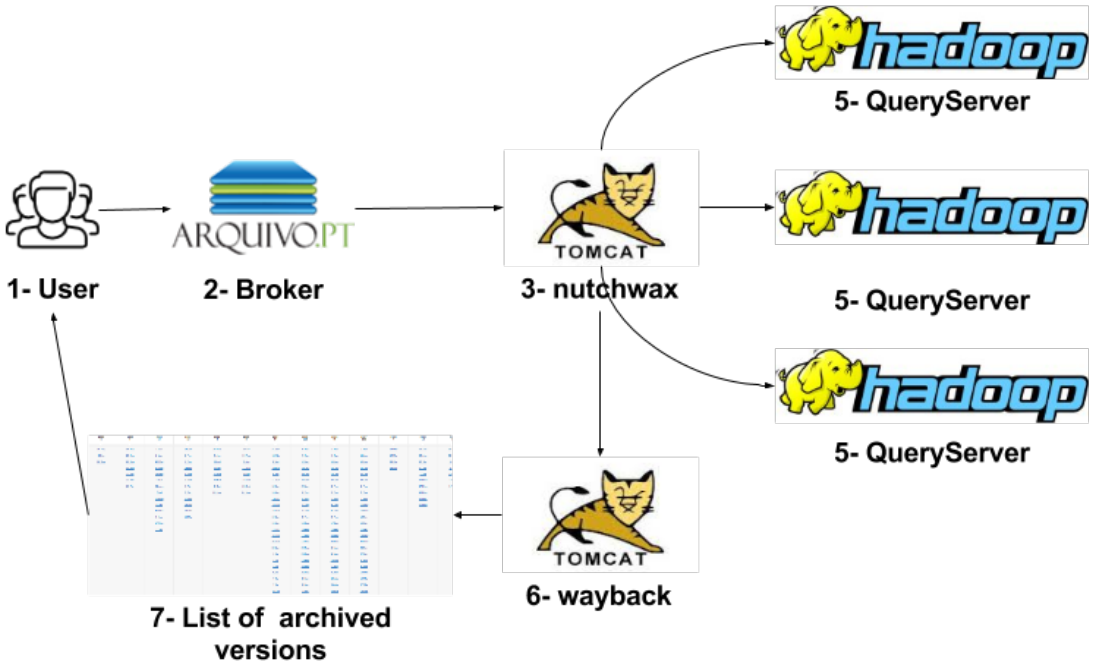
4. Each QueryServer: Returns a response that contains a posting list with the 10k most relevant results. (see figure 4.3)

5. Inside Broker: The nutchwax accesses the responses from QueryServers and performs the list of documents in XML format. The search results list is returned to the user.

4.2.2 URL search

Fig. 4.4 illustrates the workflow when a URL search query is submitted to *Arquivo.pt*.

Figure 4.4: Workflow for a URL search.



Websites on the internet does not host the same entry page name and users does not know every entry page. Thereby, exacturl was developed to improve user's experience and it expands an URL with the most common entries pages.

For searching by URL, *Arquivo.pt* provides two ways of expanding an URL:

exacturl

exacturlexpand

The exacturlexpand was added by *Arquivo.pt* team, in order to expand more types of entries pages. For instance, a entry page named index.php or index.asp is expanded with exacturlexpand and it is not with exacturl. (2)

IN: "query: www.fccn.pt"

OUT: List of archived versions form "www.fccn.pt"

1. User submits a query to Apache HTTP server.
   <http://arquivo.pt/search.jsp?l=pt&query=www.sapo.pt&btnSubmit=Pesquisar>

2. Inside Broker: Apache HTTP server redirects the request to nutchwax.

3. Inside Broker: nutchwax processes the request and redirects it to wayback with the url encoded.
   <http://arquivo.pt/nutchwax/opensearch?query=exacturlexpand%3Ahttp%3A%2F%2Fwww.fccn.pt>

4. Inside Broker: The wayback processes and redirects the request to nutchwax.

5. Inside Broker: nutchwax encodes the URL using the MessageDigest algorithm with Base32. The table 4.6 details the request parameters broadcasted to the QueryServers. (This custom encoding is necessary because NutchAnalysis will not let through '?' or '=' characters in clause values and the '&' character can't be passed in a query string because it'll confuse request.getParameter.)

6. Inside Broker: The nutchwax accesses the responses from QueryServers and performs the list of URL versions in XML format.

7. Inside Broker: The nutchwax redirects the list of URL version to wayback.

4.2.3 Page replay

Fig. 4.2 illustrates the workflow when a page replay is requested to *Arquivo.pt*.

IN: http://arquivo.pt/wayback/20130330011529/http://www.fccn.pt/
IN: Arcoffset:39093130
IN: Arcname: FCCN-PT-HISTORICAL-ia400125.20090108013027.arc.gz
OUT: Archived content

1. User submits the URL to Apache HTTP server.
   <http://arquivo.pt/wayback/20130330011529/http://www.fccn.pt/>

2. Inside the Broker: Apache HTTP redirects the request to wayback.

Table 4.6: Parameters broadcasted across QueryServers (URL search).

Parameter name	Example	Description
query	"date:19960101000000-20141231235959" exacturlexpand:"7S22KTVSC23FWOIQKE3ALEW2O4" exacturlexpand:"WU4334YSN4GZJXSXSK5ONOXPFI" exacturlexpand:"3IR4HDOD4K4UXEJKCAOSXFTFSI" exacturlexpand:"HYRDACAOIGG5CBHXGIETSYAQMI" exacturlexpand:"LT2VIFELMDO4F3YOT7KQIPDT74" exacturlexpand:"EB7TK4VBMP2ASBIK3IPZSCHU2A" exacturlexpand:"ZONLPBFHM5WNTWFEZPX336DZPQ" exacturlexpand:"6SIR575MUOXCMD6O7BR6C4ZWLM" exacturlexpand:"JVKH55INJAERJZ2AN4EJ2IMZJ4" exacturlexpand:"NQ2J7PJ6ZPJXPM2E4YXUQUD5G4" exacturlexpand:"5WQF6BFZ22VQCMMHD5L6RT7COY" exacturlexpand:"JWANRQSTMTZ2EJOITIQ2BYYTCM" exacturlexpand:"PJUVQPSJH5EUJZMPDSCIAL6WIU" exacturlexpand:"WE54IJNBGP5FOHPXHAMFECZT4U" exacturlexpand:"FZLAMUVUM2IFIGCNYQAME7H6XE" exacturlexpand:"DTHXCKDH7ZCZUCWQNAE2D2PNM4" exacturlexpand:"ZFTHMM44OMLVSWAJVGG2NPVB6I" exacturlexpand:"WBMFHC3ZXTYAIP3MKS7VESTT7E" exacturlexpand:"FLA7WJJ73FU6UU6EQ2QXZAWJHI" exacturlexpand:"NYMEII2SLVYYM2WKCJ55A7RV6E" exacturlexpand:"UI4C4WF6I67NI5CYAZ5MTWQ2F4" exacturlexpand:"4NDT7NAA4YZ4LENSYHEQKEWCAU"	sets encoded query string
numHits	20	sets the number of listed results.
searcherMaxHits	10000	sets the max number of results for the posting list.
maxHitsPerDup	2	sets the max number of repeat results on the posting list.
dedupField	site	sets the maxHitsPerDup field filter.
sortField	null	sets the field name to sort (i.e sort by date ..).
reverse	false	sets the sort order by ascending or descending.
rankingfunctions	49=0.5939482, 37=0.34503713, 45=1.2592824, 34=0.023322608	sets the ranking features and its weights.
maxHitsPerVersion	1	selects max number of repeated version.
liveAddress	[p21.arquivo.pt/193.136.192.175:21136]	sets the hostname and ipaddress about the Broker.

3. Inside the Broker: The wayback redirects the request to nutchwax. The request contains the exacturlexpand and the timestamp.

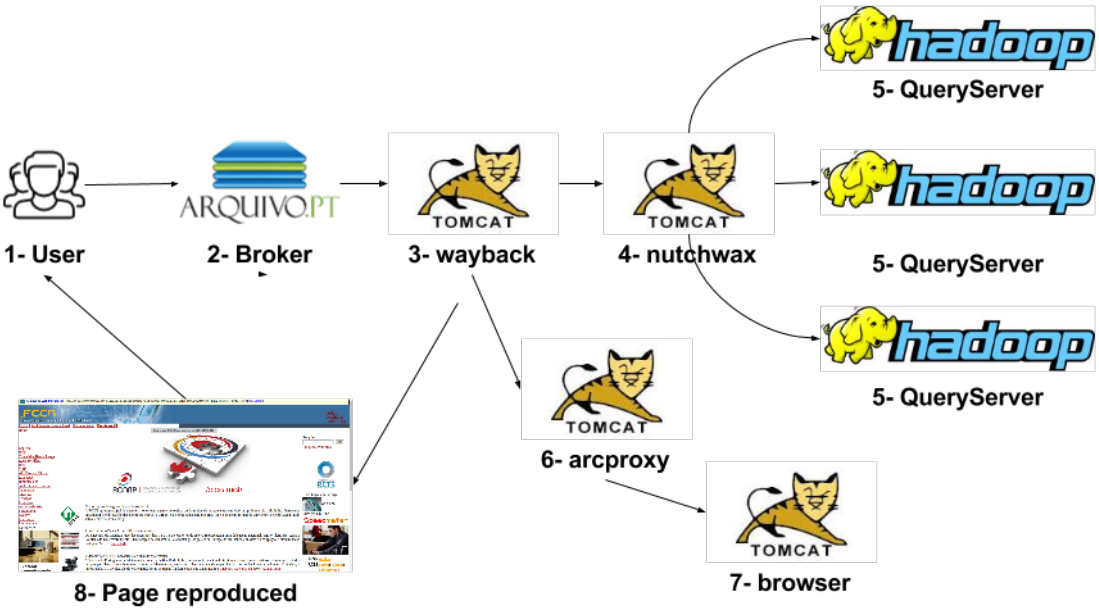
4. Broker communication with QueryServer: The nutchwax broadcasts the request with exacturlexpand and timestamp across QueryServers.

5. Inside the Broker: The nutchwax process the response from QueryServer and respond with the ARC filename and the offset to wayback.

6. Inside the Broker: The wayback requests the web content using HTTP Range Requests (51) to arcproxy. The request contains the arcfile name and the arcoffset.
<"http://arquivo.pt/arcproxy/arcproxy/FCCN-PT-HISTORICAL-ia400125.20090108013027.arc.gz",
"Content-Range: bytes 39093130 - ">

7. Inside the Broker: The arcproxy accesses BerkeleyDB and gets the host-

Figure 4.5: Workflow for a page replay.



name of the DocumentServer, that stores the arcfile.

8. Inside the Broker: The arcproxy requests the web content to browser and then streams it to wayback.

9. Inside the Broker: The wayback stops the communication when the bytes streamed from arcproxy reached the content-length of the requested web content.

4.2.4 Advanced search

The advanced search is similar with full-text search, though with a nuance you can perform your filter as you need. Hence, the fig. 4.2 illustrates also the workflow for an advanced search.

IN: "With these words: sapo" Between: 20070101000000-20141231235959 Show the pages in the format: pdf With this address: www.sapo.pt
OUT: List of documents containing the filter choices

1. User submits the advanced query to Apache HTTP server which contains the performed fields .
   <"http://arquivo.pt/search.jsp?l=pt&adv_and=sapo&adv_phr=&adv_

not=&dateStart=01%2F01%2F2007&dateEnd=31%2F12%2F2014&sort=relevance&format=pdf&site=www.sapo.pt&hitsPerPage=10&btnSubmitBottom=Pesquisar+no+Arquivo">

2. On Broker: Apache HTTP redirects the request to nutchwax.

3. On Broker: nutchwax broadcasts the requests across QueryServers. Table 4.7 details the request parameters.

Table 4.7: Parameters broadcasted across QueryServers (advanced search).

Parameter name	Example	Description
Query	sapo site:"www.sapo.pt" type:"pdf" date:"20070101000000-20141231235959"	set the query string with the performed filter.
numHits	20	sets the number of listed results.
searcherMaxHits	10000	sets the max number of results for the posting list.
maxHitsPerDup	2	sets the max number of repeat results on the posting list.
dedupField	site	sets the maxHitsPerDup field filter.
sortField	null	sets the field name to sort (i.e sort by date ..).
reverse	false	sets the sort order by ascending or descending.
rankingfunctions	49=0.5939482, 37=0.34503713, 45=1.2592824, 34=0.023322608	sets the ranking features and its weights.
maxHitsPerVersion	1	selects max number of repeated version.
liveAddress	[p21.arquivo.pt/193.136.192.175:21136]	sets the hostname and ipaddress about the Broker.

4. Each QueryServer: Returns back a response that contains a posting list with the 10k most relevant results.

5. Inside Broker: The nutchwax responds a ranked list of documents.

4.2.5 URL syntax to reference a web content

Arquivo.pt provides two manners for accessing an URL. Lucene DocId syntax and the standarized wayback syntax, the follow are examples of both respectively:

DocId `http://arquivo.pt/wayback/wayback/id24689index25`

Wayback `http://arquivo.pt/wayback/wayback/19980205082901/http://www.caleida.pt/saramago/`

Lucene DocId syntax, the number *24689* is the DocId and the number *25* is the collection entry on search-servers.txt (i.e. for instance FAWP11). Wayback

syntax 19980205082901 is the timestamp for the web content. The format of the timestamp is 1-14 digits (YYYYMMDDhhmmss).

Chapter 5

Maintenance Procedures

This chapter describes procedures for maintaining *Arquivo.pt* search system system. It is divided in the following sections:

- Indexing collection;
- Installing the *Arquivo.pt* search system;
- Deploying collection;
- Removing archived web content from the search results;
- Looking up the Lucene-index with Luke;
- Installing informative site (Arquivo.pt_CMS);

5.1 Indexing a collection

Arquivo.pt produces three types of collections, which are broad crawls (AWP), daily crawls (FAWP) and extraordinary crawls (EAWP).

AWP crawl done every 4 months;

FAWP crawl done every day;

EAWP crawl done in special occasions.

The software architecture of *Arquivo.pt* includes a Hadoop cluster (19 Computers) which is a computational cluster designed specifically for storing and

analyzing huge amounts of unstructured data in a distributed computing environment. The Hadoop cluster is used to index the harvest of web content and instructions for compiling and installing is available in (21) and (22). *Arquivo.pt* developed a bash script to execute all jobs and it is named **index-collection.sh**. The steps of the script are explained in the subsection 5.1.2. (3)

5.1.1 Indexing steps

Arquivo.pt contains a Hadoop job for the process of indexation (23).

The Hadoop job contains the following sub-jobs:

fawp_prepare: Creates a auxiliary DB with all of the gathered URLs. It is used when there are URls with the same domain name but with different timespan (only for FAWP crawl).

import: Ingests ARCs writing ARC Record parse as Nutch FetcherOutputFormat.

update: Update dbs with recent imports. This process creates the crawldb directory on the HDFS. The crawldb maintains information on all known URLs (fetch schedule, fetch status or metadata) .

invert: Invert links (create structure outlink->inlinks). This process creates the linkdb directory on the HDFS. The linkdb maintains an inverted link map, listing incoming links for each url .

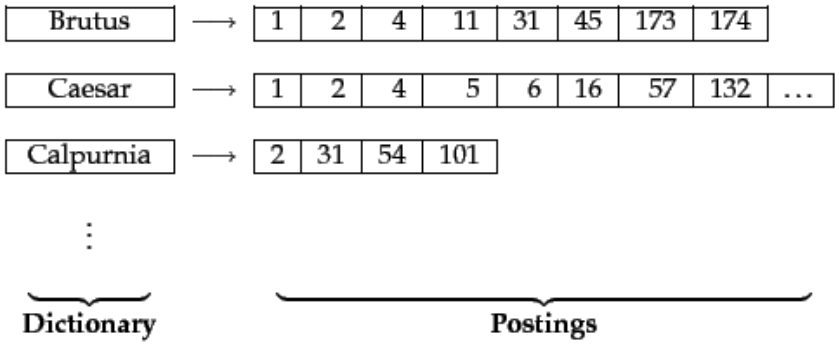
index: This process creates the indexes directory on the HDFS.

merge: Merge indexes partitions into one. This process creates the index directory on the HDFS.

index_prunning: The posting lists of the five fields (content, title, anchor, url and host) are pruned.

Arquivo.pt removes posting that are not shown in results, in order to produce a much smaller index that returns identical results. Each item in the list - which records that a term appeared in a document - is conventionally called a posting. The list is called posting list. Fig. 5.1 illustrates an example of a posting and posting list.

Figure 5.1: The two parts of an inverted index. The Dictionary is commonly kept in memory, with pointers to each posting list, which is stored on disk. (Retrieved from `http://nlp.stanford.edu/IR-book/html/htmledition/an-example-information-retrieval-problem-1.html`)



The posting list of *Arquivo.pt* contains five fields:

- content
- title
- anchor
- url
- host

5.1.2 Workflow

1. The index-collection.sh creates a auxiliary DB with all URLs and sets up the Hadoop cluster for supporting collections types with multiple URLs (only for FAWP). This guarantees one Hadoop job per URL.

2. The index-collection.sh takes all the URLs from the DocumentServer and adds them to the crawldb. As a central part of Nutch, the crawldb maintains information on all known URLs.

3. Based on the data of crawldb, index-collection.sh creates a fetchlist and places it in a newly created segment directory.

4. The fetcher gets the content of the URLs on the fetchlist and writes it back to the segment directory. This step usually is the most time-consuming one.

5. Before indexing, all the links need to be inverted, the inverted links are saved in the linkdb.

6. Using data from all possible sources (crawldb, linkdb and segments), the indexer creates an index and saves it within the index directory.

7. Pruning Indexes, the posting lists of the five fields (content, title, anchor, url and host) are pruned. All documents on the posting lists of mime types not searchable are discarded. Fields stored not on index can also be discarded using the "-del" option followed by the field names. Execute the command without parameters to see the help description.

5.2 Installing the *Arquivo.pt* search system

This section joins all needed instructions for setting up the *Arquivo.pt* search system.

- required software and file system architecture appendix A
- guidance to set up the search system (24) (25)
- code location (26)
- code compilation (27)

5.3 Deploying a collection

This section provides a guidance about the process of deploying a collection to the search service.

Fig. 5.2 shows how the process of deploying a collection evolves, starting from the stored indexes until it gets accessible. (28) (29) (30) (31)

1. **Setting up the QueryServer**

   Table 5.3 contains the bundle of files/folders for a QueryServer installation.

Figure 5.2: QueryServer flowchart for deploying a collection

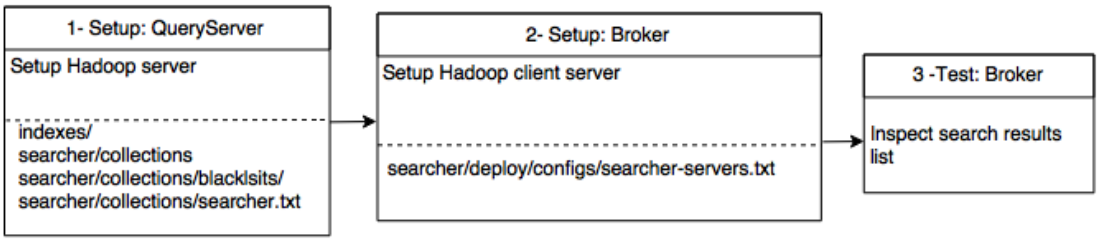


Table 5.1: Bundle of files necessary for deploying indexes (QueryServer).

Path	**Observations**
/indexes/	the location of the indexes.
/collections/search-server.txt	contains the hosted collections on QueryServer.
/collections/Blacklist/	contains the Lucene DocID removed from the results search list.

(a) Copy indexes and segments to the folder *indexes/*.

(b) In bash execute the binary to stop QueryServer:

collections/stop-slave-searchers.sh .

(c) Add a new entry with the name collection into file search-server.txt.

Table 5.2 is an entry example of search-server.txt with the collection AWP5.

Table 5.2: Example of QueryServer: search-servers.txt

Host Name	Port Name	Lucene Index Path	Hadoop heap size(MB)	Lucene Blacklist Path
p57.arquivo.pt	10015	indexes/outputs_AWP5	15000	collections/blacklists/AWP5.txt

(d) In bash execute the binary to start the QueryServer:

collections/start-slave-searchers.sh.

2. **Setting up the Broker**

Add a new entry in *deploy/config/search-server.txt* with the new collection, in this case AWP5.

Table 5.4 is an example of a file configuration for Broker.

Table 5.4: Example of Broker: search-servers.txt

Machine_Name	Port_Name
p57.arquivo.pt	10015

Table 5.3: Bundle of files necessary for deploying indexes (Broker).

Path	Observations
/deploy/configs/search-servers.txt	contains the location of the indexes supplied by each QueryServer.
/deploy/current/	contains the location of the web applications.
/arcproxy/	contains the arcproxy data base.
/dictionaries/	contains the indexes built for the query suggestion.
/run/	contains the tomcat pid file.
/scripts/	contains the binary to generate the arcproxy DB.

3. **Test Broker**

   For checking up that the indexes were properly set up, you have to query the system with a full-text or URL search and the search results list should be returned. Figure 5.3 illustrates the search results list for the queries *fccn* and *fccn.pt* respectively.

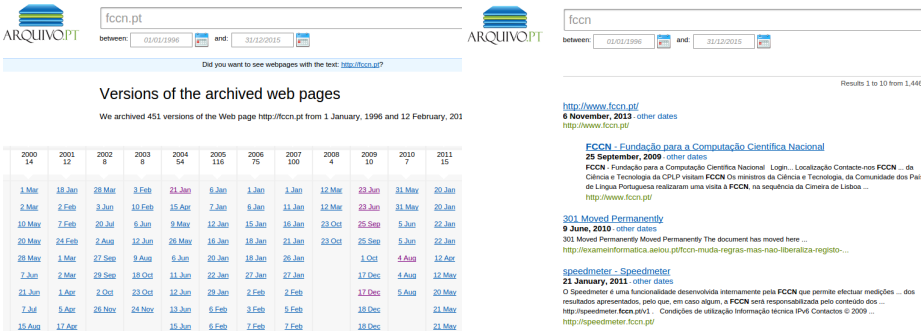
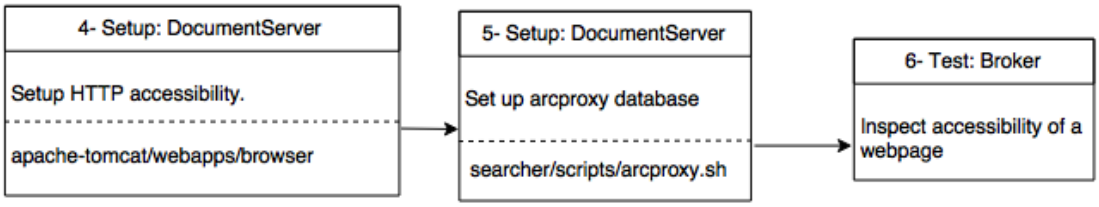


Figure 5.3: Search results list when the indexes were properly set up.

4. **Setting up the DocumentServer:** HTTP accessibility

Figure 5.4: DocumentServer flowchart for deploying a collection



   (a) Create symbolic links to arcfiles into tomcat webapp *browser*.

   (b) Start tomcat;

(c) Guarantee that the collection is acessable through a HTTP:8080, i.e `p70.arquivo.pt:8080/browser/files/AWP5`;

5. **Setting up the DocumentServer:** ArcProxy database

   (a) Delete old files in *collection/arc_file_AWP5.txt* that might contain previous configurations;

   (b) In bash run the binary code illustrated in table 5.5.

Table 5.5: Bash script to set up the ArcProxy

Script name	Collection path	http request for browser.war	Broker server
scripts/arcproxy.sh	-d collections/AWP5/	-u http://p70.arquivo.pt:8080/ browser/files/AWP5	p58.arquivo.pt

6. **Test Broker**

   For assessing the set up of the DocumentServer, you have to request a reply of a web content. Fig. 5.5 is page reply for the archived web content `http://arquivo.pt/wayback/20140930090138/http://www.fccn.pt/pt/index.php`.

5.3.1 Understanding the Broker search-servers.txt

This section underlines that the entry order on search-servers.txt have to be coherent by means of a cumbersome mistakes. Table 5.6 details an example of a search-servers.txt on a Broker.

Table 5.6: Excerpt from search-servers.txt on P58.arquivo.pt (Broker).

QueryServer	Port_Name	Index Position	Collection Name
p55.arquivo.pt	21111	0	IA
p55.arquivo.pt	21112	1	Roteiro
p55.arquivo.pt	21113	2	BN
p55.arquivo.pt	60008	7	FAWP8
p56.arquivo.pt	10011	8	AWP11
p56.arquivo.pt	10012	9	AWP12
p57.arquivo.pt	10001	19	AWP1
p57.arquivo.pt	10003	20	AWP3
p57.arquivo.pt	10004	21	AWP4

Table 5.7 details an example of a search-servers.txt on a QueryServer.

Figure 5.5: Page replay of a DocumentServer

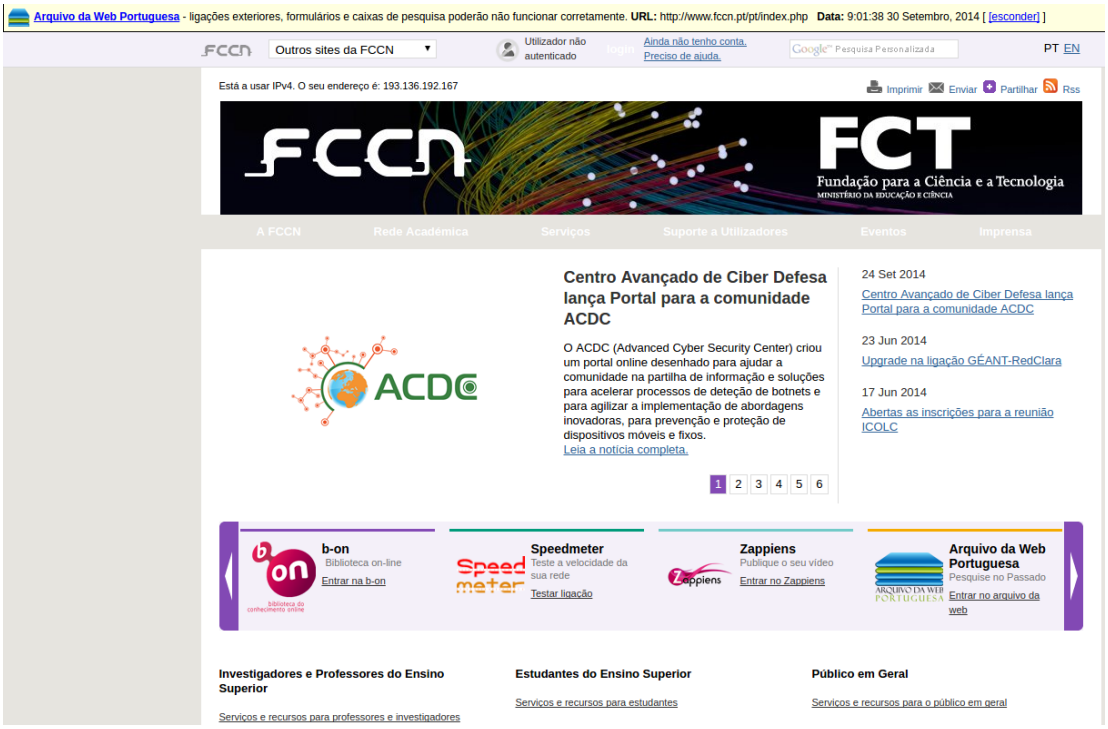


Table 5.7: Excerpt from search-servers.txt on p55.arquivo.pt (QueryServer).

QueryServer	Port_Name	Indexes	Java heap memory	Blacklist	Index Position	Collection Name
p55.arquivo.pt	21111	indexes/outputs_IA	17300	collections/blacklists/empty.txt	0	IA
p55.arquivo.pt	21112	indexes/outputs_Roteiro	100	collections/blacklists/empty.txt	1	Roteiro
p55.arquivo.pt	21113	indexes/outputs_BN	2000	collections/blacklists/empty.txt	2	BN

Hence, collection named *IA* is the first entry of search-servers.txt(table 5.6), which means that on search-servers.txt(table 5.7) the entry must also be the first.

Table 5.8 details a search-server.txt which does not match with the search-servers.txt on table 5.6. The first position on QueryServer is the collection *Roteiro* and on Broker is collection IA.

Table 5.8: Excerpt from wrongly search-servers.txt (QueryServer).

QueryServer	Port_Name	Indexes	Java heap memory	Blacklist	Index Position	Collection Name
p55.arquivo.pt	21112	indexes/outputs_Roteiro	100	collections/blacklists/empty.txt	0	Roteiro
p55.arquivo.pt	21113	indexes/outputs_BN	2000	collections/blacklists/empty.txt	1	BN
p55.arquivo.pt	21111	indexes/outputs_IA	17300	collections/blacklists/empty.txt	2	IA

As a result, we could be displayed a misplaced web content. Fig 5.6 is an

illustration of a search-server.txt wrongly created.

(a) IA Collection (Pos. 0) (b) Roteiro Collection (Pos. 2)

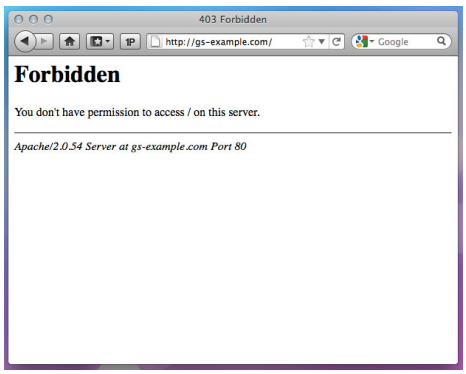
Figure 5.6: Result of a search-server.txt with a non coherent order.

5.4 Removing archived web content from the search

Arquivo.pt has two approaches for removing an index from the search result list.

Forbid access to archive web content: Apache HTTP redirects. Still, Apache HTTP only sets the index inaccessible to user, whereby it will be visible in the search result lists. Fig. 5.7 illustrate the returned Apache page to users;

Figure 5.7: Removing access with Apache HTTP rules



Remove from the search results list: removes the access to a exact Lucene DocID. It sets the index to null, therefore there are not a matching index to a web content. Whereupon, the web content will not be displayed either

on the search results list or in format of web content. The web content is removed from both search results list shown on fig. 5.3; (32) (33);

5.5 Full-text search with Luke

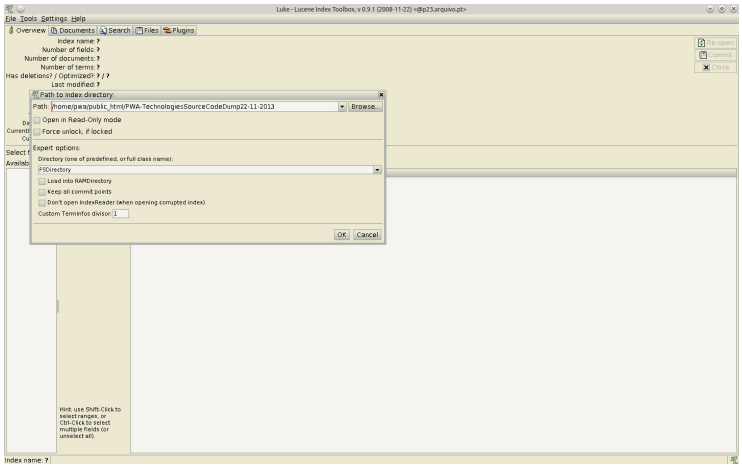
Arquivo.pt made use of version 0.9.1 of Luke to do a full-text search on the generated indexes. Luke is a handy development and diagnostic tool, which accesses already existing Lucene indexes and allows you to display and modify their content.

The following describes how to search by URL using Luke.

1. Download Luke. (34)
2. Launch Luke: java -jar lukeall-0.9.1.jar.

   Fig. 5.8 presents and example of an user invoking Luke.

   Figure 5.8: Screenshot of first page of Luke



3. Select the path where the indexes are stored.
4. Select tab Search
5. Select the field exacturl
6. Get the expanded and encoded URL using opensearch API.

   `http://p21.arquivo.pt/opensearch?query=exacturlexpand:http://www.sapo.ua.pt/`

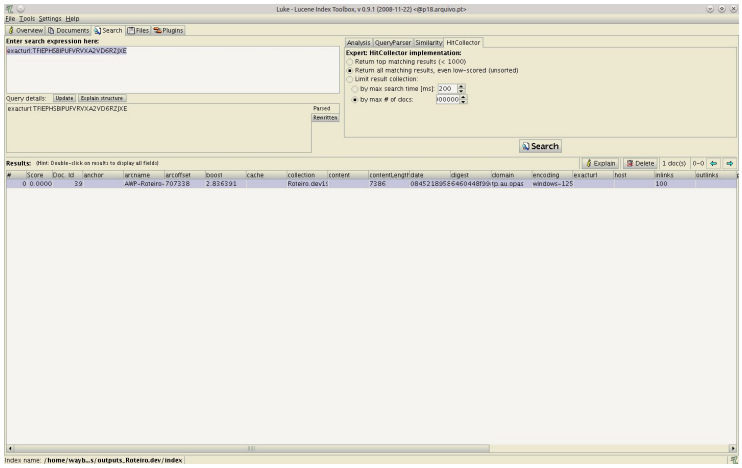
7. Fig. 5.9 presents the chosen exacturl: *TFIEPHSBIPUFVRVXA2VD6RZJXE.*

Figure 5.9: Opensearch header result for the query http://sapo.ua.pt.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rss xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/" xmlns:pwa="http://arqui
version="2.0">
<channel>
<title>PWA Search Engine</title>
<description>PWA search results for query: exacturlexpand:VOQHVNHZ5EOCL4YVE3PXFIN5NE
exacturlexpand:TFIEPHSBIPUFVRVXA2VD6RZJXE exacturlexpand:SN6KO65F7V47EUBQUI24RIPGIA
exacturlexpand:SLDLIHHLUTCBPOPWFTRMZF7OQQ exacturlexpand:SUXZOJRAPJQST7LR4IKSORXSTY
exacturlexpand:BY3O5L3DFFRHYCGFZEZY24CBMI exacturlexpand:FQA6T7DMGAGOYIK4UKT3IEEFMU
exacturlexpand:ZZ2EFZF4FOQFFX224CDZL6CJCY exacturlexpand:ZQJCMF46SQHNBJNJQAQ5DHGYCU
exacturlexpand:AR4BQJPZPZIU4B55CLJ7FX4VDE exacturlexpand:HJUSS62JKBC2UC2SQRHPAN7NZI
<link>http://archive.pt</link>
<opensearch:totalResults>8</opensearch:totalResults>
```

8. On tab Analysis select KeywordAnalyser
9. On tab Hitcollector change to: "Return all matching results, even low-scored"
10. On the top right side insert: exacturl:TFIEPHSBIPUFVRVXA2VD6RZJXE
11. Press search and the result will printed out. The fig 5.10 presents the final result.

Figure 5.10: Searchinf for the URL http://sapo.ua.pt with Luke.



5.6 Installing the informative site (Arquivo.pt_CMS)

For a newly installation of Plone there are many plugins that is necessary to be installed and configured. (35) (36) (37)

The *sobre.arquivo.pt* installation of Plone, contains several scripts that are running in background, which are:

zopePloneApacheBackup.sh backups the `sobre.arquivo.pt` every day and replicates the master version to its slave.

verificapagina.sh warns *Arquivo.pt* team when there are changes at `sobre.arquivo.pt`.

5.7 Monitoring the *Arquivo.pt* service

Arquivo.pt has the following tools to guarantee the best service quality.

Nagios: (Internal) monitors the infrastructure. `sirens.fccn.pt/nagios/`

Uptimerobot: (External) monitors the infrastructure. `https://uptimerobot.com/dashboard#mainDashboard`

Awstats: analysis the logs of Arquivo.pt. `logs.arquivo.pt/awstats/awstats.pl?config=arquivo.pt`

Awstats: analysis the logs of *sobre.arquivo.pt*. `logs.arquivo.pt/awstats/awstats.pl?config=arquivo-web.fccn.pt`

Google analytics: tracks and reports the *Arquivo.pt* and *sobre.arquivo.pt* traffic `analytics.google.com/analytics/web/?authuser=0#home/a21825027w43114507p47220271/`

Selenium tests: tests for the web application. `github.com/arquivo/pwa-technologies/tree/master/functional-tests`

Ansible scripts: automates the IT infrastructure. `gitlab.fccn.pt/awp-arquivo-da-web-portuguesa/ansible/tree/master`

Ganglia: is a scalable distributed system monitor tool for high-performance computing systems such as clusters and grids. `http://stats.arquivo.pt`

Chapter 6

Customizing Full-text of Search Ranking

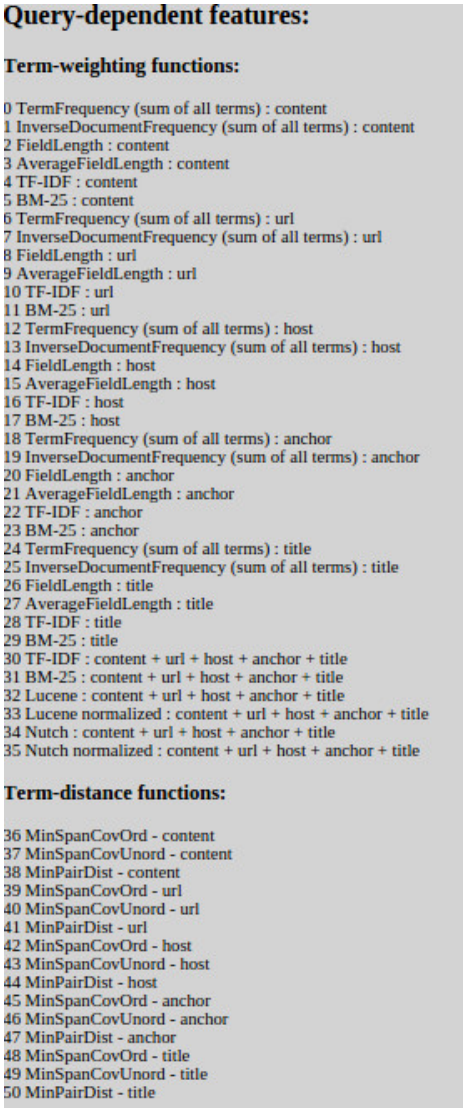
The ranking system orders by relevance millions of documents matching query. It uses several heuristics to compute a relevance value for each document matching a query, and presents the documents sorted by these values (40). On the Internet, users expect no more than a few seconds for a response. Hence, several ranking functions were added, such as query-dependent, query-independent and time-aware (42). Fig. 6.4 is the user interface developed for testing combinations of features and heights. For collaborators of *Arquivo.pt* it is available in `http://arquivo.pt/searchTests.jsp`.

Figure 6.1: Webpage provided for testing ranking features

6.1 Query-dependent features

Query-dependent features depends both on the contents of the document and the query. Query-dependent models estimate the document's relevance according to a given query.

Figure 6.2: searchTests.jsp: Query-dependent features.



6.1.1 Term-weighting functions

Term-weighting functions features estimate the similarity between the query and the different sections of a document version(anchor text of incoming links, text body, title, and URL) (41).

Term Frequency (TF) Term Frequency look at term t in document d, which counts the number of times that t occurs in d. Suppose we have a collection of N documents. Define $\{f_i, j\}$ to be the frequency (number of occurrences) of term (word) i in document j. Then, define the term frequency $\{TF_i, j\}$ to be:

$$TF_{i,j} = \frac{f_{i,j}}{max_k f_{kj}} \tag{6.1}$$

That is, the term frequency of term i in document j is $\{f_i, j\}$ normalized by dividing it by the maximum number of occurrences of any term (perhaps excluding stop words) in the same document. Thus, the most frequent term in document j gets a TF of 1, and other terms get fractions as their term frequency for this document (48).

Inverse Document Frequency (IDF) The IDF component acts to discriminate between informative and non-informative query terms. Those terms that have a high IDF are considered more informative, because they rarely occur in the collection. On the other hand, terms that have a low IDF are considered uninformative, since they occur in many documents. As the number of documents in a collection increases, IDF becomes increasingly important in order to discriminate between those documents that contain non-informative query terms and those that contain high informative query terms (48).

The IDF for a term is defined as follows. Suppose term i appears in $\{n_i\}$ of the N documents in the collection. Then

$$IDF_i = log(\frac{N}{n_i}) \tag{6.2}$$

TF-IDF is the combination of this two algorithms. Consider a document

containing 100 words wherein the word *Arquivo.pt* appears 3 times. The term frequency (i.e., tf) for *Arquivo.pt* is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word *Arquivo.pt* appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12. (38)

Algorithm 6.1: TF-IDF (4)

TF-IDF
```
public PwaTFxIDF(Vector<Integer> tf, Vector<Integer> idf, int nTerms, int nDocs) {
        double tfNorm;
        double idfNorm;

        for (int i=0;i<tf.size();i++) {
                if (tf.get(i)!=0) {
                        tfNorm=(double)tf.get(i)

                        idfNorm=Math.log((double)nDocs/idf.get(i));
                        score+= tfNorm*idfNorm;
                }
        }
}
```

BM-25 The relevance weighting model, also known as RSJ by the name of its creators(Roberston and Sparck-Jones), has been one of the most influential model in the history of Information Retrieval. It is a probabilistic model of retrieval that tries to answer the following question:

What is the probability that this document could be relevant to this query?

'Query' is a particular instance of an information need, and 'document' a particular content description. The purpose of this question is to rank the documents in order of their probability of relevance according the Probability Ranking Principle (39).

Algorithm 6.2: BM25 (5)

```
public PwaBM25(Vector<Integer> tf, Vector<Integer> idf, int nTerms, double avgNTerms,
int nDocs, double k1Parameter, double bParameter) {
        double tfNorm;
        double idfNorm;

        for (int i=0;i<tf.size();i++) {
                if (tf.get(i)!=0) {
                        idfNorm=Math.log10((double)(nDocs-idf.get(i)+0.5) / (idf.get(i)+0.5));
                        tfNorm=(double)(tf.get(i)*(k1Parameter+1)) /
                        (tf.get(i)+k1Parameter*((1-bParameter)+bParameter*((double)nTerms/avgNTerms)));
```

```
                        score+= idfNorm*tfNorm;
                    }
                }
            }
```

Lucene Lucene combines Boolean model (BM) of Information Retrieval with Vector Space Model (VSM) of Information Retrieval - documents "approved" by BM are scored by VSM.

Algorithm 6.3: Lucene Similarity (6)

```
public PwaLuceneSimilarity(Vector<Vector<Integer>> tfPerField, Vector<Vector<Integer>>
idfPerField, Vector<Integer> nTermsPerField, int nDocs) {
        double tfNorm;
        double idfNorm;
        Vector<Integer> tf;
        Vector<Integer> idf;
        int nTerms=0;
        double sumOfSquaredWeights=0;

        for (int j=0;j<tfPerField.size();j++) {
                tf=tfPerField.get(j);
                idf=idfPerField.get(j);
                nTerms=nTermsPerField.get(j);

                for (int i=0;i<tf.size();i++) {
                        if (tf.get(i)!=0) {
                                tfNorm= Math.sqrt((double)tf.get(i));
                                idfNorm= 1+Math.log((double)nDocs/(double)(idf.get(i)+1));
                                sumOfSquaredWeights+= Math.pow(idfNorm, 2);
                                idfNorm= Math.pow(idfNorm, 2);
                                score+= tfNorm * idfNorm * norm(PwaIndexStats.FIELDS[j],nTerms,boosts[j]);

                        }
                }
        }

        score*= queryNorm(sumOfSquaredWeights);
}
```

Algorithm 6.4: Lucene Similarity Normalized (7)

```
public PwaLuceneSimilarityNormalized(Vector<Vector<Integer>> tfPerField, Vector<Vector<Integer>>
 idfPerField, Vector<Integer> nTermsPerField, int nDocs)
{
        similarity=new PwaLuceneSimilarity(tfPerField, idfPerField, nTermsPerField, nDocs);
        score=similarity.score();
        score=MAX_SCORE-score;
        score=(float)Math.pow(Math.E, -1*score/MAX_SCORE);
}
```

Nutchwax Lucene but with a different normalization by field length was a small variation of this function used in NutchWAX, with a different normalization by field length (44).

Algorithm 6.5: Nutchwax Similarity (8)

```
public PwaNutchSimilarity(Vector<Vector<Integer>> tfPerField,
 Vector<Vector<Integer>> idfPerField, Vector<Integer> nTermsPerField, int nDocs) {
        double tfNorm;
        double idfNorm;
        Vector<Integer> tf;
        Vector<Integer> idf;
        int nTerms=0;
        double sumOfSquaredWeights=0;

        for (int j=0;j<tfPerField.size();j++) {
                tf=tfPerField.get(j);
                idf=idfPerField.get(j);
                nTerms=nTermsPerField.get(j);

                for (int i=0;i<tf.size();i++) {
                        if (tf.get(i)!=0) {
                                tfNorm= Math.sqrt((double)tf.get(i));
                                idfNorm= 1+Math.log((double)nDocs/(double)(idf.get(i)+1));
                                sumOfSquaredWeights+= Math.pow(idfNorm, 2);
                                idfNorm= Math.pow(idfNorm, 2);
                                score+= tfNorm * idfNorm * norm(PwaIndexStats.FIELDS[j],nTerms,boosts[j]);

                        }
                }
        }

        score*= queryNorm(sumOfSquaredWeights);
}
```

Algorithm 6.6: Nutchwax Similarity Normalized (9)

```
public PwaNutchSimilarityNormalized(Vector<Vector<Integer>> tfPerField,
Vector<Vector<Integer>> idfPerField, Vector<Integer> nTermsPerField, int nDocs) {
        similarity=new PwaNutchSimilarity(tfPerField, idfPerField, nTermsPerField, nDocs);
        score=similarity.score();
        score=MAX_SCORE-score;
        score=(float)Math.pow(Math.E, -1*score/MAX_SCORE);
}
```

6.1.2 Term-distance functions

Term-distance features use the distance between terms in the different sections of a document version to quantify the relatedness between them, such as the Minimal Span Weighting function (41). Farther information available in `https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/PwaPositionsManager.java`.

Span is defined as the length of the shortest document segment that covers all query term occurrences in a document, including repeated occurrences (49).

Minimum coverage (MinCover) MinCover is defined as the length of the shortest document segment that covers each queryterm at least once in a

document (49).

Minimum Pair Distance (MinDist) The minimum pair distance is defined as the smallest distance value of all pairs of unique matched query terms. Formally,

$$MinDist = min_{q_1,q_2 \epsilon Q \cap D, q_1 \neq q_2} \{Dis(q_1, q_2; D)\} \tag{6.3}$$

(49).

Average Pair Distance (AveDist) The average pair distance is defined as the average distance value of all pairs of unique matched query terms. Formally,

$$AveDist = \frac{2}{n(n-1)} \sum_{q_1,q_2 \epsilon Q \cap D, q_1 \neq q_2} \{Dis(q_1, q_2; D)\} \tag{6.4}$$

, where n is the number of unique matched query terms in D, and in the sum, we count Dis$(q_1, q_2; D)$ and Dis$(q_2, q_1; D)$ only once (49).

Term-distance ranking features are based on the above.

MinSpanCovUnord (Minimum Span Coverage Unordered) minimum span including all query terms.

MinSpanCovOrd (Minimum Span Coverage Ordered) minimum span including all query terms ordered as submitted.

MinSpanPairDist (Minimum Span Pair Distance) minimum distance between two query terms.

Algorithm 6.7: Computing Distances (10)

```
public void computeDistances(int doc) throws IOException {
        minSpanCovUnordered=Integer.MAX_VALUE;
        minSpanCovOrdered=Integer.MAX_VALUE;
        minPairDist=Integer.MAX_VALUE;

        if (terms==null || terms.size()<2) {
                minSpanCovUnordered=0;
                minSpanCovOrdered=0;
                minPairDist=0;
                return;
        }
```

```
for (int i=0;i<terms.size();i++) {
        positions[i]=terms.get(i).getPos(doc);
        if (positions[i]==null) {
                return;
        }
}

queue.clear();
int end = 0;
boolean done = false;

for (int i=0;i<positions.length;i++) {
        if (!positions[i].next()) {
                return;
        }
        if (positions[i].get()>end) {
                end=positions[i].get();
        }
        queue.put(positions[i]);
}

do {
        PwaPositions pos = (PwaPositions)queue.pop();
        int start = pos.get();
        int next = ((PwaPositions)queue.top()).get();
        for (int i=start; i<=next && !done; ) {
                start = i;
                if (!pos.next()) {
                        done=true;
                }
                else {
                        i=pos.get();
                }
        }


        int matchLength = end-start - nTermsInQuery+1;
        if (minSpanCovUnordered>matchLength) {
                minSpanCovUnordered=matchLength;
        }
        if (minSpanCovOrdered>matchLength) {
                boolean testOrder=true;
                for (int i=1;i<positions.length && testOrder;i++) {
                        int a = (pos==positions[i-1]) ? start : positions[i-1].get();
                        int b = (pos==positions[i]) ? start : positions[i].get();
                        if (a>b || (offsetTerms!=null && offsetTerms.get(i)-offsetTerms.get(i-1)!=b-a)) {
                                testOrder=false;
                        }
                }
                if (testOrder) {
                        minSpanCovOrdered=matchLength;
                }
        }
        if (minPairDist>next-start-1) {
                minPairDist=next-start-1;
        }

        if (minSpanCovOrdered==0) {
                done=true;
        }
        if (!done) {
                if (pos.get()>end) {
                        end=pos.get();
                }
```

```
                    queue.put(pos);
                }
            }
            while (!done);
}
```

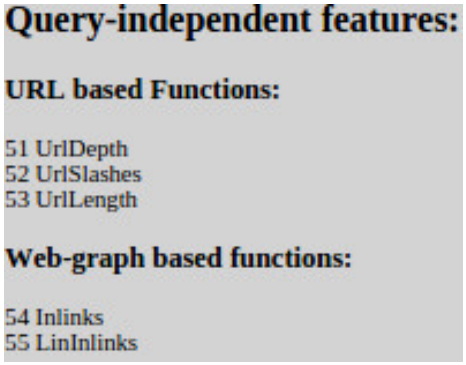
Algorithm 6.8: Minimum Span (11)

```
public PwaMinSpan(int span) {
        score=(float)Math.log(1+Math.pow(Math.E, -1*span));
}
```

6.2 Query-Independent Features

Query-Independent Features, which depend only on the document, but not on the query. Therefore, these models rank documents according to an importance, quality or popularity measure computed independently of the query (41).

Figure 6.3: searchTests.jsp: Query-independent features.



URL features compute an importance measure based on the probability of URLs representing an entry page, using the number of slashes, their length, or if they refer to a domain, sub-domain or page (41).

Web-graph features estimate the popularity or importance of a document version inferred from the graph of hyperlinks between versions. These features include the number of inlinks to a version (41).

The following is the explanation of each query-independent feature:

Url Depth A URL defines the unique location of a document on the WWW. It consists of the name of the server, a directory path and a filename. A

first observation is that documents that are right at the top level of a specific server (i.e. the documents of which the URL is no more than a server name) are often entry pages. Also, as we descend deeper into the directory tree, the relative amount of entry pages decreases. Thus, the probability of being an entry page seems to be inversely related to the depth of the path in the corresponding URL.(47)

Algorithm 6.9: Url Depth (12)

```
public PwaUrlDepth(String surl) throws MalformedURLException {
        URL url=new URL(surl);
        String urlParts[]=surl.split("/");
        int c=2; // http + ''
        if (urlParts.length==1+c) {
                score=3;
                return;
        }
        if (urlParts[urlParts.length-1].indexOf('.')!=-1) { // file
                score=0;
                return;
        }
        if (urlParts.length==2+c) {
                score=2;
                return;
        }
        score=1;
}
```

Url Slashes number of URL slashes. (47)

Algorithm 6.10: Url Slashes (13)

```
public PwaUrlSlashes(String surl) throws MalformedURLException {
        score=0;
        for (int i=7; i<surl.length(); i++) {
                if (surl.charAt(i)=='/') {
                        score++;
                }
        }
}
```

Inlinks A hyperlink is a connection between a source and a target document. Our inlinks based prior is based on the observation that entry pages tend to have a higher number of inlinks than other documents (i.e. they are referenced more often). (47)

LinInlinks a Linearization of the number of inlinks .(47)

Algorithm 6.11: LinInlinks (14)

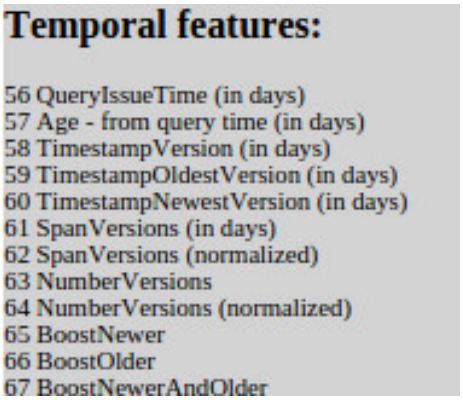
```
public PwaLinInlinks(int nInlinks) {
        if (nInlinks==0) {
```

```
                score=0;
        }
        else {
                score=(float)Math.log10(Math.pow(nInlinks, POWER));
        }
}
```

6.3 Temporal features

The temporal features consider the time dimension of the web. They include the age of a document version and the features about lifespan based on the long-term persistence of web documents.

Figure 6.4: searchTests.jsp: Temporal features.



QueryIssueTime Query issue time in days (42)

Age - from query time Timespan in days from the query issue time to the version date (42)

Algorithm 6.12: Age (15)

```
public PwaAge(long docTimestamp, long queryTimestamp) {
        float diff = queryTimestamp-docTimestamp;
        score = diff / DAY_MILLISEC;
}
```

TimestampVersion Age of the version in days (42)

TimestampOldestVersion Age of the oldest version of the same URL in days (42)

TimestampNewestVersion Age of the newest version of the same URL in days (42)

SpanVersions) Days between the oldest and newest version of the same URL (42)

Algorithm 6.13: Java code for SpanVersions feature (16)

```
public PwaSpanVersions(long maxTimestamp, long minTimestamp, long maxSpan) {
        long diff = (maxTimestamp-minTimestamp) / (long)DAY_MILLISEC;
        if (maxSpan==0)  {
                score=0;
        }
        else if (diff==0) {
                score=0;
        }
        else {
                score = (float)Math.log10(diff) / (float)Math.log10(maxSpan);
        }
}
```

SpanVersions (normalized) Normalized days between the oldest and newest version of the sam URL (42)

NumberVersions Number of versions of the same URL (42)

Algorithm 6.14: NumberVersions (17)

```
public PwaNumberVersions(long numberVersions, long maxNumberVersions) {
        score = (float)Math.log10(numberVersions) / (float)Math.log10(maxNumberVersions);
}
```

NumberVersions (normalized) Normalized number of versions of the same URL (42)

BoostNewer Exponential decay of the age of the version that boosts more recent versions (42)

BoostOlder Exponential decay of the age of the version that boosts more older versions (42)

Algorithm 6.15: BoostOlder (18)

```
public PwaBoostOlder(long docTimestamp, long maxTimestamp, long minTimestamp) {
        float maxSpan=maxTimestamp-minTimestamp;
        maxSpan/= DAY_MILLISEC; // span in days
        float span=docTimestamp-minTimestamp;
        span/= DAY_MILLISEC; // span in days
        score=(float)Math.pow(Math.E, -1*span/maxSpan);
}
```

BoostNewerAndOlder Exponential decay of the age of the version that boosts more recent and older versions (42)

Algorithm 6.16: BoostNewerAndOlder (19)

```
public PwaBoostNewerAndOlder(long docTimestamp, long maxTimestamp, long minTimestamp) {
        long middle=(maxTimestamp+minTimestamp)/2;
        if (docTimestamp<middle) {
                PwaBoostOlder ranker=new PwaBoostOlder(docTimestamp, maxTimestamp, minTimestamp);
                score=ranker.score();
        }
        else {
                PwaBoostNewer ranker=new PwaBoostNewer(docTimestamp, maxTimestamp, minTimestamp);
                score=ranker.score();
        }
}
```

6.4 Ranking system used in the *Arquivo.pt*

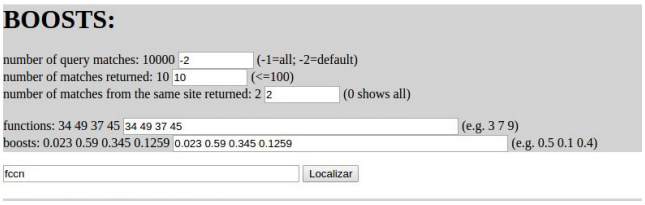
Arquivo.pt developed a ranking system base on L2R algorithm in order to find the best features. The project is base on Lucene, but with improvements for instance on the Lucene features. PwaLucene is available in `https://github.com/arquivo/pwa-technologies/tree/master/PwaLucene`, since you have PwaLucene installed the configuration of wax-default.xml is need, in order to support all the features added. This file contains the ranking features which the system are using to rank. The following frame illustrates the file wax-default.xml, which contains the ranking features that *Arquivo.pt* has in production (`https://github.com/arquivo/pwa-technologies/blob/master/PwaArchive-access/projects/nutchwax/conf/wax-default.xml`).

<name> ranking.functions </name>
<value>**34** 0.023 **49** 0.59 **37** 0.345 **45** 1.259</value>
<description>
</description>

The features are imported to the system through `https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/PwaFunctionsWritable.java`, and afterwards this functions will be send over protocol RPC for ranking documents. The backbone of the L2R is located in `https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/PwaScorerFeatures.java`, where the whole process of ranking are processed.

The fig. 6.5 illustrates the searchTests.jsp set up with the actual features of *Arquivo.pt*. This webpages page provides an interface for assess all of the combination of *Arquivo.pt* features.

Figure 6.5: Boosting new features



The table 6.1 details the ranking features that are in production.

Table 6.1: Ranking features on production in *Arquivo.pt*

Feature number	**Name of the features**	**Fields**	**Weight**
34	Nutch	content + url + host + anchor + title	0.023
49	MinSpanCovUnord	title	0.593
37	MinSpanCovUnord	content	0.345
45	MinSpanCovOrd	anchor	1.259

6.5 Future Work

Arquivo.pt has developed a test collection named PWA9609 for our own purpose (`https://github.com/arquivo/pwa-technologies/wiki/TestCollection`). A L2R dataset for researching learning to rank applied to WAIR was created. This dataset named L2R4WAIR, more information available in `https://github.com/arquivo/pwa-technologies/wiki/L2R4WAIR`. The top 6 most important ranking features which were achieved for the temporal-dependent are:

- BM25 over all fields
- TD-IDF over all fields
- Number of versions of a URL
- TF-IDF over the hostname of URL
- Length of the shortest text with all query terms in title
- Days between the first and last versions of a URL

Appendix A

Software Requirements

This appendix details the required files, applications and configurations for setting up the *Arquivo.pt* search system. Further information are available in `https://github.com/arquivo/pwa-technologies/wiki/Install` and `http://wiki.priv.fccn.pt/Install_Search_Environment`.

Requirements of Broker

/home/wayback/searcher/ : project root;

apache-tomcat-8.0.30/ : contains the Apache Tomcat binaries;

arcproxy/ : contains the BarkeleyDB;

dictionaries/indexIA : contains the dictionaries for spellchecker.war;

deploy/current : contains web applications;

deploy/configs/ : contains the search-servers.txt;

run/ : contains the pids of running applications;

scripts/ : contains the arcproxy.sh script;

Apache HTTP : Apache HTTP.

Requirements of DocumentServer

/opt/searcher/ : project root;

apache-tomcat-8.0.30/ : contains the Apache Tomcat binaries;

collections/ : contains collections that are on this DocumentServer;

run/ : contains the pids of running applications.

Requirements of QueryServer

/home/wayback/searcher/ : project root;

hadoop-search-servers : contains the HadoopRPC server;

run/ : contains the pids of running applications;

collections/ contains QueryServer configurations;

collections/blacklists/ contains Lucene index that are removed from search results;

../indexes/ contains indexes;

/etc/init.d/hadoopserver is the binary to start the QueryServer.

Version of the current Arquivo.pt installation:

NutchWax 0.11.0

PWA_Lucene 1.0.0 (extensions of lucene-2.1.0)

Wayback 1.2.1

Tomcat 8.0.30

Hadoop-DEV 0.14.5

Maven 3.x

ANT 1.7.1

JAVA SE 7.x

Plone 3.0.3

LinguaPlone 2.0

qPloneComments 3.1

PloneGazette 3.0.0

Appendix B

Mime-types indexed

The configuration file that contains information about the mime-types indexed is at `https://github.com/arquivo/pwa-technologies/blob/master/PwaArchive-access/projects/nutchwax/nutchwax-webapp/target/nutchwax-webapp-0.11.0-SNAPSHOT/WEB-INF/classes/mime-types.xml`.

The following list details which are the set up mime-types:

application/andrew-inset

application/javas

application/mac-binhex40

application/mac-compactpro

application/msword

application/oda

application/dfp

application/postscript

application/smil

application/vnd.mif

application/vnd.ms-excel

application/vnd.ms-powerpoint

application/vnd.oasis.opendocument.presentation

application/vnd.oasis.opendocument.presentation-template

application/vnd.oasis.opendocument.spreadsheet

application/vnd.oasis.opendocument.spreadsheet-template

application/vnd.oasis.opendocument.text

application/vnd.oasis.opendocument.text-template

application/vnd.oasis.opendocument.text-master

application/vnd.oasis.opendocument.text-web

application/vnd.sun.xml.calc

application/vnd.sun.xml.calc.template

application/vnd.sun.xml.impress

application/vnd.sun.xml.impress.template

application/vnd.sun.xml.writer

application/vnd.sun.xml.writer.template

application/vnd.wap.wbxml

application/vnd.wap.wmlc

application/vnd.wap.wmlscriptc

application/xhtml+xml

application/x-bzip2

application/x-bcpio

application/x-cdlink

application/x-chess-pgn

application/x-cpio

application/x-csh

application/x-director

application/x-dosexec

application/x-dvi

application/x-futuresplash

application/x-gtar

application/x-gzip

application/x-hdf

application/x-javascript

application/x-kword

application/x-kspread

application/x-kpresenter

application/x-kchart

application/x-killustrator

application/x-koan

application/x-latex

application/x-netcdf

application/x-ogg

application/x-rar-compressed

application/x-rpm

application/x-sh

application/x-shar

application/x-shockwave-flash

application/x-stuffit

application/x-sv4cpio

application/x-sv4crc

application/x-tar

application/x-tcl

application/x-tex

application/x-texinfo

application/x-troff

application/x-troff-man

application/x-troff-me

application/x-troff-ms

application/x-ustar

application/x-wais-source

application/zip

audio/basic

audio/midi

audio/mpeg

audio/x-aiff

audio/x-mpegurl

audio/x-pn-realaudio

audio/x-pn-realaudio-plugin

audio/x-realaudio

audio/x-wav

chemical/x-pdb

chemical/x-xyz

image/bmp

image/gif

image/ief

image/jpeg

image/photoshop

image/png

image/tiff

image/vnd.djvu

image/vnd.wap.wbmp

image/xcf

image/x-cmu-raster

image/x-portable-anymap

image/x-portable-bitmap

image/x-portable-graymap

image/x-portable-pixmap

image/x-rgb

image/x-xbitmap

image/x-xpixmap

image/x-xwindowdump

message/news

message/rfc822

model/iges

model/mesh

model/vrml

text/css

text/html

text/plain

text/richtext

text/rtf

text/sgml

text/tab-separated-values

text/vnd.wap.wml

text/vnd.wap.wmlscript

text/xml

text/x-setext

video/mpeg

video/quicktime

video/vnd.mpegurl

video/x-msvideo

video/x-sgi-movie

x-conference/x-cooltalk

Bibliography

[1] I. Archive. nutchwax-home page. http://archive-access.sourceforge.net/, 06 Nov 2007.

[2] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaArchive-access/projects/nutchwax/nutchwax-core/src/main/java/org/archive/access/nutch/NutchwaxQuery.java, year = 2016, note =.

[3] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/scripts/index-collection.sh, year = 2016, note =.

[4] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/features/querydependent/PwaTFxIDF.java, 2016. [Online; accessed 04-Fev-2016].

[5] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/features/querydependent/PwaBM25.java, 2016. [Online; accessed 04-Fev-2016].

[6] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/features/querydependent/PwaLuceneSimilarity.java, 2016. [Online; accessed 04-Fev-2016].

[7] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/

features/querydependent/PwaLuceneSimilarityNormalized.java, 2016. [Online; accessed 04-Fev-2016].

[8] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/features/querydependent/PwaNutchSimilarity.java, 2016. [Online; accessed 04-Fev-2016].

[9] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/features/querydependent/PwaNutchSimilarityNormalized.java, 2016. [Online; accessed 04-Fev-2016].

[10] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/PwaPositionsManager.java, 2016. [Online; accessed 04-Fev-2016].

[11] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/PwaLucene/src/main/java/org/apache/lucene/search/features/querydependent/PwaMinSpan.java, 2016. [Online; accessed 04-Fev-2016].

[12] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/queryindependent/PwaUrlDepth.java, 2016. [Online; accessed 04-Fev-2016].

[13] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/queryindependent/PwaUrlSlashes.java, 2016. [Online; accessed 04-Fev-2016].

[14] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/queryindependent/PwaLinInlinks.java, 2016. [Online; accessed

04-Fev-2016].

[15] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/temporal/PwaAge.java, 2016. [Online; accessed 04-Fev-2016].

[16] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/temporal/PwaSpanVersions.java, 2016. [Online; accessed 04-Fev-2016].

[17] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/temporal/PwaNumberVersions.java, 2016. [Online; accessed 04-Fev-2016].

[18] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/temporal/PwaBoostOlder.java, 2016. [Online; accessed 04-Fev-2016].

[19] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/caf1beea7f0f005a4b95218259d4ffa2ba2f4823/PwaLucene/src/main/java/org/apache/lucene/search/features/temporal/PwaBoostNewerAndOlder.java, 2016. [Online; accessed 04-Fev-2016].

[20] Arquivo.pt. https://github.com/arquivo/pwa-technologies/blob/master/Plone/Plone-3.0.5-UnifiedInstaller.tar.gz, 2016. [Online; accessed 04-Fev-2016].

[21] Arquivo.pt. https://github.com/arquivo/pwa-technologies/wiki/Install, 2016. [Online; accessed 04-Fev-2016].

[22] Arquivo.pt. http://wiki.priv.fccn.pt/Indexacao, 2016. [Online; accessed 04-Fev-2016].

[23] Arquivo.pt. https://github.com/arquivo/pwa-technologies/tree/master/PwaArchive-access/projects/nutchwax/nutchwax-core/src/main/java/org/archive/access/nutch/jobs, 2016. [Online; accessed 04-Fev-2016].

[24] Arquivo.pt. http://wiki.priv.fccn.pt/Install_Search_Environment, 2016. [Online; accessed 04-Fev-2016].

[25] Arquivo.pt. https://github.com/arquivo/pwa-technologies/wiki/Install, 2016. [Online; accessed 04-Fev-2016].

[26] Arquivo.pt. https://github.com/arquivo/pwa-technologies/, 2016. [Online; accessed 04-Fev-2016].

[27] Arquivo.pt. https://github.com/arquivo/pwa-technologies/wiki/Compile, 2016. [Online; accessed 04-Fev-2016].

[28] Arquivo.pt. http://wiki.priv.fccn.pt/Recolhas, 2016. [Online; accessed 04-Fev-2016].

[29] Arquivo.pt. http://wiki.priv.fccn.pt/Tuning_memory_for_Index, 2016. [Online; accessed 04-Fev-2016].

[30] Arquivo.pt. http://wiki.priv.fccn.pt/Indexacao#Colocar_em_Produ.C3.A7.C3.A3o, 2016. [Online; accessed 04-Fev-2016].

[31] Arquivo.pt. http://wiki.priv.fccn.pt/Hardware_do_Tomba, 2016. [Online; accessed 04-Fev-2016].

[32] Arquivo.pt. https://github.com/arquivo/pwa-technologies/wiki/Operation#blacklist, 2016. [Online; accessed 04-Fev-2016].

[33] Arquivo.pt. http://wiki.priv.fccn.pt/BlackList_Info, 2016. [Online; accessed 04-Fev-2016].

[34] Arquivo.pt. http://sobre.arquivo.pt/~pwa/PWA-TechnologiesSourceCodeDump22-11-2013/lukeall-0.9.1.jar, 2016. [Online; accessed 04-Fev-2016].

[35] Arquivo.pt. https://github.com/arquivo/pwa-technologies/wiki/Install#install-plone, 2016. [Online; accessed 04-Fev-2016].

[36] Arquivo.pt. http://wiki.priv.fccn.pt/Plone, 2016. [Online; accessed 04-Fev-2016].

[37] Arquivo.pt. http://wiki.priv.fccn.pt/Zope_e_Apache, 2016. [Online; accessed 04-Fev-2016].

[38] Arquivo.pt. http://www.tfidf.com/, 2016. [Online; accessed 04-Fev-2016].

[39] E. D. Buccio and G. M. D. Nunzio. Are there new bm25 expectations? In R. Basili, F. S. 0001, and G. Semeraro, editors, *IIR*, volume 964 of *CEUR Workshop Proceedings*, pages 1–12. CEUR-WS.org, 2013.

[40] M. Costa. Sidra: a flexible web search system. Master's thesis, University of Lisbon, Faculty of Sciences, November 2004. Also available as Technical Report DI/FCUL TR 4-17.

[41] M. Costa. *Information Search in Web Archives*. PhD thesis, Faculty of Sciences of the University of Lisbon, December 2014.

[42] M. Costa, F. Couto, and M. Silva. Learning temporal-dependent ranking models. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 757–766, New York, NY, USA, 2014. ACM.

[43] M. Costa, J. Miranda, D. Cruz, and D. Gomes. Query Suggestion for Web Archive Search. Technical report, Foundation for National Scientific Computing, 01 2012.

[44] M. Costa and M. J. Silva. Evaluating web archive search systems. In X. S. Wang, I. Cruz, A. Delis, and G. Huang, editors, *Proceedings of the 13th International Conference on Web Information Systems Engineering*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, November 2012.

[45] D. Gomes, M. Costa, D. Cruz, J. a. Miranda, and S. a. Fontes. Creating a billion-scale searchable web archive. In *Proceedings of the 22nd international conference on World Wide Web companion*, WWW '13 Companion, pages 1059–1066, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[46] D. Gomes, D. Cruz, J. Miranda, M. Costa, and S. Fontes. Creating a searchable web archive. Technical report, Foundation for National Scientific Computing, 05 2012.

[47] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in*

Information Retrieval, SIGIR '02, pages 27–34, New York, NY, USA, 2002. ACM.

[48] A. Rajaraman. *Mining of Massive Datasets*. Cambridge University Press, Cambridge, 2011.

[49] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 295–302, New York, NY, USA, 2007. ACM.

[50] B. Tofel. `https://github.com/arquivo/pwa-technologies/blob/master/PwaArchive-access/projects/wayback/dist/src/site/xdoc/administrator_manual.xml`, 2016. [Online; accessed 04-Fev-2016].

[51] w3.org. `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`, 2016. [Online; accessed 04-Fev-2016].