WARC PORTAL V 1.0

Administrator Documentation

Table of Contents

Preamble

" || > " indicates a command to be inputted in a bash shell.

The University of Alberta and its researchers have a collection of Web Archives, but do not have an easy way to analyze them. The WARC portal project aims to deal with extracting, searching and analyzing web archive files. The project provides intuitive and easy access for researchers to browse and search through thousands of documents, and provides tools for analyzing their collections. These tools include an array of searches and filters, as well as helpful visualizations of their data by analyzing keywords used across the collections. WARC Portal provides an invaluable tool for experienced digital humanities and social science researchers. It presents the web archive data in an intuitive way that will help researchers find overall patterns and trends.

Features of WARC Portal:

- Searching through archived webpages
- Searching through archived images
- Displaying webpages in original format
- Text and Image analysis

Initial Authors:

Cheng Chen: cheng10@ualberta.ca

Adriano Marini: marini@ualberta.ca

Kevin Tang: tkevin@ualberta.ca

Mate Verunica: verunica@ualberta.ca

Download / Access

The initial authors will have transferred ownership of a private GitHub repository to an administrator. You may clone or download it from this repository. For information on how to use GitHub, please see https://help.github.com/

System Requirements

Hardware

OS: Linux (preferably Ubuntu 16.04 or greater)
Storage: 500+ GB Free Space
Recommendation: **RAM:** 8GB Minimum (32GB Preferred)

Dependencies

- Scala Language Support < https://www.scala-lang.org/ >
- Oracle Java JDK < http://www.oracle.com/technetwork/java/index.html >
- Apache Spark < http://spark.apache.org/ >
- Node / NPM < https://nodejs.org/en/ >
- Apache Maven < https://maven.apache.org/ >

Installation

PostgreSQL

```
|| > sudo apt-get update
|| > sudo apt-get install postgresql postgresql-contrib
```

Upon installation Postgres is set up to use "ident" authentication, meaning that it associates Postgres roles with a matching Unix/Linux system account. If a Postgres role exists, it can be signed in by logging into the associated Linux system account.

The installation procedure created a user account called postgres that is associated with the default Postgres role. In order to use Postgres, we'll need to log into that account. You can do that by typing:

```
|| > sudo -i -u postgres
```

You will be asked for your normal user password and then will be given a shell prompt for the postgresuser.

From the postgres Linux account, you have the ability to log into the database system. However, we're also going to demonstrate how to create additional roles. The postgres Linux account, being associated with the Postgres administrative role, has access to some utilities to create users and databases.

Taken from https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-14-04 on 03-12-2016 12:39PM

WARCBASE

Warcbase is an open-source platform for managing web archives built on Hadoop and HBase. The platform provides a flexible data model for storing and managing raw content as well as metadata and extracted knowledge. Tight integration with Hadoop provides powerful tools for analytics and data processing via Spark.

The first and most common is to analyze web archives using Spark: these functionalities are contained in the warcbase-core module, and what WARC Portal uses as its analysis basis.

Steps to Installation

```
|| > git clone http://github.com/lintool/warcbase.git
|| > cd warcbase
|| > mvn clean package
```

BUILD SUCCESS indicates you are now able to use warcbase.

```
|| > wget http://archive.apache.org/dist/spark/spark-
1.6.1/spark-1.6.1-bin-hadoop2.6.tgz
|| > tar xzf spark-1.6.1-bin-hadoop2.6.tgz

|| > cd spark-1.6.1-bin-hadoop2.6.tgz
```

You can then run Spark Shell with the (recommended) command:

```
|| > ./bin/spark-shell --driver-memory 4G --jars
~/warcbase/warcbase-core/target/warcbase-0.1.0-SNAPSHOT-
fatjar.jar
```

(Assuming you cloned warcbase in ~/ -- If not, replace the directory in the above command)

Once this is complete, you can do fun analytics in spark shell if you wish.

WARC Portal needs Warcbase (and spark shell) to perform analytics, so ensure that all of the scripts (see: scripts.readme)
point to the correct location of spark shell.

If you would like to read more about warcbase: http://lintool.github.io/warcbase-docs/Getting-Started/

PYWB

pywb is an open-source library used by WARC Portal to display the webpages
located in WARC files.

SETUP

The best time to complete this would be after you have installed WARC Portal.

```
|| > sudo apt-get install build-essential libssl-dev libffi-dev
python-dev
```

** APT-GET is exclusive to Ubuntu. If you are using a different linux flavor, please search for and install these packages through alternate means **

```
|| > git clone https://github.com/ikreymer/pywb.git
```

** Before continuing, please go into the WARC-Portal git directory, and refer to the files 'default_banner.js' and 'wb.css' in the pywb sub directory. Please use these files to replace the existing files in '../pywb/pywb/static/' to allow for design integration into the WARC Portal user interface **

```
|| > cd pywb

|| > python setup.py install
```

INITIAL SETTINGS

After PYWB has successfully installed, basic configuration is needed to allow it to function.

> change directory into the area where you prefer to store files <

```
|| > mkdir wayback_collection

|| > cd wayback_collection

|| > wb-manager init warc_portal

|| > wb-manager add warc_portal /path/to/warcs/*.warc.gz

|| > nohup wayback &
```

** (Ensure you are running Wayback in the wayback_collection top folder) **

After this, you can navigate to www.yoururl.com:8080 to search through the elements located in the WARC archives.

ADDING WARCs TO WAYBACK

The scripts will automatically add new files to the system for usage in the system.

WARC Portal

Django API (First)

CD into WARC-Portal folder

```
|| > virtualenv venv

|| > source venv/bin/activate

|| > pip install -r requirements.txt
```

React UI (Second)

CD into WARC-Portal Folder

```
|| > npm install

|| > npm run build-prod

|| > node app/server.js
```

Server will be bound to localhost:5000

Editing scripts (Third)

There are four main script files in the system that perform all of the backend functions that are needed for data population. As they are system specific, they will need some editing before they will work correctly in your environment

…/WARC-Portal/scripts/run_spark.sh
…/WARC-Portal/scripts/info_get.sh
…/WARC-Portal/scripts/tf_get.sh
…/WARC-Portal/scripts/clean.sh
…/WARC-Portal/scripts/cron.bak

All of these files contain a fixed path to the file trees we used during development. Therefore, they must be edited to reflect your system's organization. Please replace all file paths with the correct paths for your system. It would be best to convert all paths to new paths with the same conversion.

Executing the CRON Job (Fourth)

```
|| > crontab cron.bak
```

This will install the contents of cron.bak as your user account's cron document. This means that the linux subsystem will automatically run the tasks described at fixed intervals, which in our case, automates the WARC file processing functionality of the system.

Code Documentation

REACT UI

https://facebook.github.io/react/docs/hello-world.html

General Flow

Component / Page --> Saga --> API --> Saga --> Reducer --> Component/Page

auth.js
API classes for handling authentication with the Django backend

- Login deals with login into the django backend.

collection.js
API classes for communicating with the Django backend for the purposes
of getting and creating collections

- getCollections gets a list of the current user's collections
- getFiles gets a list of the files currently in the System
- postCollections creates a new collection from selected files

document.js
Classes for retrieving documents from the back end server

- getDocs works with types of documents
- getImages works on images.

filter.js
API class for retrieving filters from the Django backend

- getFilters returns all filters

App.jsx
The react component describing the body of the app.

Content.jsx
The react component describing the tabbed browsing of the main page. Dictates the navigation between the search, images, and login tabs.

DateField.jsx
React component for the date picker of the toolbar.

DocElementList.jsx
React component for the individual elements of the list on homepage.

DocumentList.jsx
React component for the feature of a document list.

- Allows updating of the list based on search queries.
- Enables pagination of results.
- Sets design and properties of the entire list.

FilterOptions.jsx
The react component that deals with the filtering options on the side
of the page. Collects all of the options, and renders them into a list, and then handles
their selection and the changes as a result of the selections.

Menu.jsx
React component for the top menu bar of the homepage which includes
the search box, and navigation buttons.

Toolbar.jsx
The filtering toolbar that allows for searching within dates / collections

URLBuilder.js
Builds a URL String from passed information for use in query strings

Collections.jsx
The react component for the collections management page. Handles rendering the collection
list, and a list of files that can be selected
and named into a new collection

Login.jsx
The react component for the login form

Images.jsx
The react class describing the image display gallery.

NotFound.jsx
Page not found landing component.

Visualizations.jsx
Renders the tf-idf graph.

WordCloud.jsx
Renders the Word Cloud visualizations.

Reducers
Reducers handle the connection between the sagas and the API, allowing for
changes in state based on the returns from the API calls.

Reducers/auth.js
Handles the state changes as a result of authentication.

Reducers/collections.js
Handles state changes as a result of obtaining collections.

Reducers/docs.js
Describes how the state of the application changes
upon success of certain functions

- State upon success of fetch list
- State upon success of fetch image
- State when onLoad is executing

Reducers/files.js
Handles state changes as a result of obtaining lists of file names.

Reducers/index.js
Describes how the state of the application changes
upon success of certain functions, dealing specifically
with calls to reducers.

Reducers/index.js/cloneObject

Clone object helper function needed to make sure the copied object is immutable
Object.assign() copies by reference when deep cloning, so we can't use it

<https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Object/assign>
Even though JSON functions don't work well with Date(), Regex() and functions,
this implementation is perfect for our needs. Redux needs the state to be serializable and sent to the redux tools,
which means that we couldn't store Date() in state even if we wanted to. Unless we screw the tools.

Parameters: object
Returns: any

Sagas
Sagas are called by react components, and rely on a reducer to send through requests
to the API and back.

Sagas/auth.js
Handles Redux flow for the API calls of authentication. Returns all necessary
information for proper authentication

Sagas/collections.js
Handles the redux flow for obtaining lists of collections and posting new collections to the database.

Sagas/documents.js
Sagas dealing specifically with documents:

- Call to fetch document list
- Call to fetch image list

Sagas/files.js
Handles the redux flow for obtaining lists of files from the database.

Sagas/index.js
Main saga generator - allows usage of sagas in the system.
Sagas allow side-effects (asynchronous calls) to occur
seamlessly.

Stylesheets
http://sass-lang.com/
SASS files that dictate the style of our webpages.

test
Built-in test components -- front end uses Selenium for
acceptance testing.

app.js
This is the master file for our REACT JS front end. Creates the middleware and stores necessary to operate the sagas. Creates the page, and defines the paths a user can take through interaction.

server.js
Dictates the server settings for the front-end.

Scripts

** All files not listed here are temporary or backup files **

cron.bak

The configuration file for jobs to be run with cron.

Each task to run is defined through a single line
indicating with different fields when the task will be run

and what command to run for the task.

WARC Portal uses cron to continuously cycle run_spark.sh.

image_ranking.scala
---
image_ranking uses warcbase's built in analytics to rank the top
images from a file.

job.scala

Goes through all of the files in a WARC, and obtains:
- URL
- Crawl Date
- Content
- A list of images related to a URL

and outputs this information to a parsable file.

reset_warc.sh

Clears out all temporary files and re-makes the directory for continued
operation.

run_spark.sh

Shell script, called by the cron job, to perform the following tasks:

- Run job.scala (process files in the warc_store directory)
- Parse the output of the job
- Intake all of that data into the database

Django / REST API

Django creates and compiles its own documentation. When the server is running, these are available at http://localhost:8000/admin/doc AND http://localhost:8000/docs

Testing

You can run our E2E tests using Nightwatch and Selenium.

```
|| > npm install -g nightwatch
```

Alternatively, if you've already ran npm install, you can access the nightwatch binary through the node_modules. Next you must update your webdrivers for selenium and chrome before running the tests found in selenium_tests.

```
|| > npm run e2e-setup
|| > nightwatch
```

License

WARC Portal

Other Packages

WARCBASE

Pywb

This program comes with ABSOLUTELY NO WARRANTY
This is free software, and you are welcome to redistribute it
under certain conditions

D3

Copyright (c) 2014-2015 Eric. S Bullington, Lim Yang Wei, and project contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

REACT

BSD License

For React software

Copyright (c) 2013-present, Facebook, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name Facebook nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Django

Copyright (c) Django Software Foundation and individual contributors.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of Django nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

POSTGRESQL

PostgreSQL Database Management System
(formerly known as Postgres, then as Postgres95)