Supreme Court Coverage & Analytics Application, Version 2.0

Administrator Manual

Written by Evan Cole

November 28, 2018

Contents

Introduction

This manual is a primer for installing, accessing, and maintaining the Supreme Court Coverage & Analytics Application (SCOTUSApp). SCOTUSApp is a collection of tools built to scrape, store, analyze, and view articles pertaining to the US Supreme Court. It consists of three parts:

- **Article Collector**– a back-end Python script that scrapes custom Google Alerts RSS Feeds, NewsAPI search results, and Supreme Court-specific pages on major news sites for relevant articles, analyzing them for text sentiment and image entities via the Google Cloud API. The script is designed to run on a timer, collecting articles multiple times throughout the day. It should never be seen by the end user.

- **SupremeCourtApp MySQL database** – where all of the scraped and analyzed data is stored. It also should never be seen by the end user, unless viewed through the web application.

- **Web application** – the front-end PHP application that enables users to view and download information from the database.

Nearly everything necessary to run the application is included in the “scotusapp” deliverable, which should be provided to you. Note – if a copy of the deliverable is downloaded from Github, it won’t contain anything sensitive like credentials, logins, or server information, nor will it have any images.

Currently, the application is run and maintained on an Amazon Web Services EC2 Instance provided by the University of Kentucky. As a result, most of the instructions in this manual are based on this platform.

Installation

Installing Libraries

SCOTUSApp has been developed on Python 3.6, PHP 7, and MySQL 5 – it is highly recommended that these versions, or more recent ones, are used to run the application. Also necessary are several external Python libraries, and should generally be installed using Python 3’s library manager, pip3 (on the EC2 instance, it is referred to as pip-3.6). Unless otherwise stated, the pip package names are the same as the name of the package.

- feedparser
- newspaper (install as newspaper3k)
- MySQLdb (install as mysqlclient)

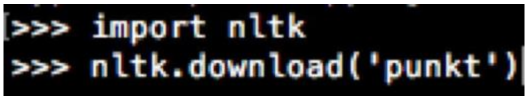
- tldextract
- PIL, aka Pillow (Python Image Library) [install as pillow]
- Google Cloud Language (install as google-cloud-language)
- Google Cloud Vision (install as google-cloud-vision)
- beautifulsoup (install as beautifulsoup4)
- requests
- Natural Language Toolkit (install as nltk)
- numpy
- scipy
- scikit-learn

It's worth mentioning that the "protobuf" library, necessary for the Google APIs, may need a specific version to be installed (also via pip) – 3.6.1 seems to work, likely others as well.

There are two notable exceptions in library installations:

- **NewsAPI** - Due to a minor issue with the requests library, the NewsAPI package had to be slightly altered; therefore, install that library from the .zip archive enclosed in the "scotusapp" deliverable, under the /install/ folder - navigate to the unzipped directory in your terminal and run the "python3 setup.py install" command to install it.

- **nltk's "punkt" module** - Newspaper requires nltk's "punkt" module in order to get article keywords. In order to install this, nltk will first need to be installed using pip. Then, from an instance of the Python 3.6 shell, the following commands will need to be issued:

  ```
  [>>> import nltk
  >>> nltk.download('punkt')
  ```

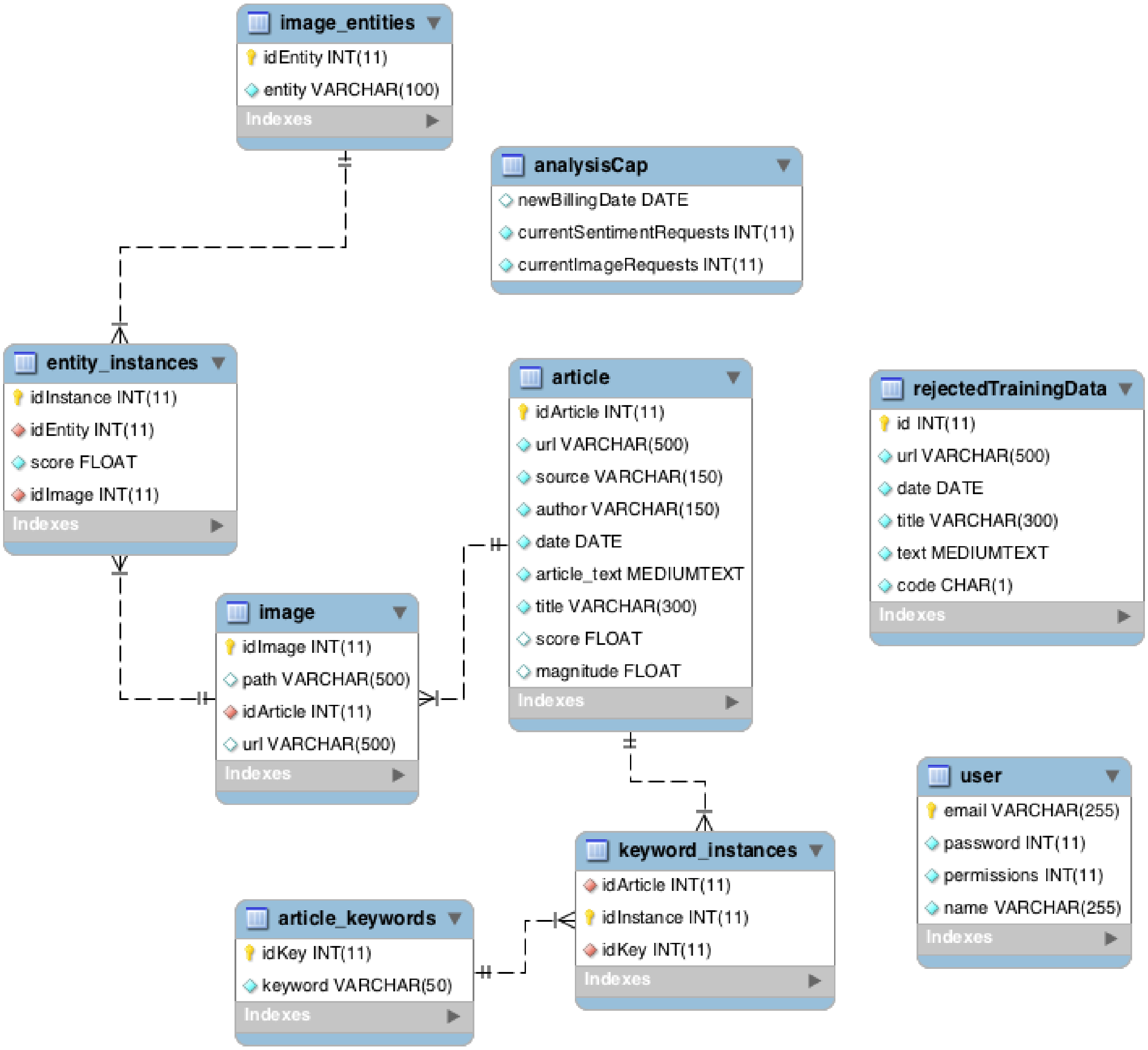
  The module should then be successfully installed.

- **nltk's "stopwords" module** – the text classifier we use to determine relevancy requires a list of English stopwords provided by nltk. Download this in the same manner as the punkt module, only with the line: nltk.download("stopwords")

Importing the Database

Under the /backups/ folder is a .sql file – this is a copy of the database to start with. It should be imported into MySQL either through the command line or an application like phpMyAdmin.

For the sake of documentation here is a diagram of the database:



EER Diagram of application database

Installing the Application

Not much work needed here – just place the entire “scotusapp” folder in the /var/www/html/ folder on the server (or, if on a local XAMPP instance, the /htdocs/ folder), and set the permissions accordingly – it is recommended that only /webapp/ and /images/ are public; everything else (but ESPECIALLY /install/ and /docs/) should be hidden from public view.

Setting Environment Variables

For security reasons, database credentials and the file path to the Google Cloud API credentials file are passed in via environment variables. Environment variables need to be set in two different locations – for the Python script, in the /etc/environment file, and for the PHP webapp, the /etc/sysconfig/httpd file. Only certain variables need to be set in each file – refer to the “environment_variable_notes.txt” file in the /docs/ folder for more information.

If any changes are made to the environment variables, they won’t be reflected until the server is rebooted with the “sudo reboot” command.

Setting the Cronjob

The cronjob is the utility we use to put the article collector on a timer, so that it collects articles several times throughout the day. In the deliverable, under /install/, should be a shell script titled run_script.sh (not publicly available). Set its permissions to 755. Then, run the command “crontab -e” and add the script to the crontab with the desired timer (look up cronjobs for more info). It is worth mentioning that the output of this script will be emailed to the administrator account upon finish.

Also included is a shell script titled backup_db.sh (also not public) – this script periodically backs up the database (minus the images). Add it to a cronjob in the same way. This script also sends a notification email upon finish.

Maintenance

Connecting to the Server

In order to connect to the server, SSH into it using the scotusapp.pem key in the /install/ folder. You will likely need to:

- move scotusapp.pem to your ~/.ssh folder

- set scotusapp.pem permissions to 600

- set ~/.ssh/ permissions to 700

More information is available in the “IMPORTANT_INFORMATION.txt” file, under /docs/.

Copying Files to the Server

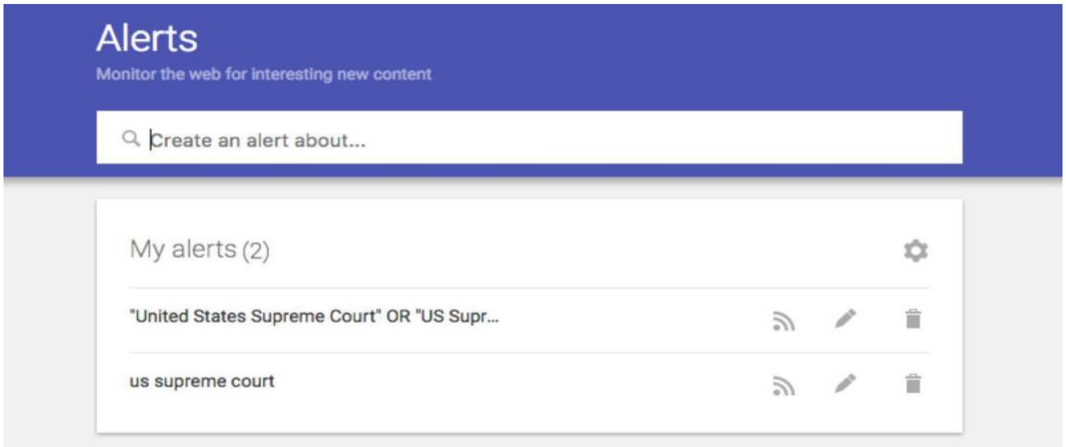
You can transfer files from your local machine to the server either by using the “scp” command in your terminal, or using an FTP program like FileZilla. However, everything must be copied from your local machine to your user’s home folder on the server, and *then* transferred to the root directories using the “sudo cp” command when you SSH in.

Administrator Account

The Google Cloud APIs and Google Alerts RSS feeds are tied to a single Gmail account. For more information, see the “IMPORTANT_INFORMATION.txt” file under /docs/.

Google Alerts RSS Feeds

The RSS feeds used to gather data are custom feeds courtesy of Google Alerts; though the feed links can be accessed by anyone, they are tied to the administrator account.



"United States Supreme Court" OR "US Supr...

us supreme court

A view of the Google Alerts website and custom RSS feeds

Awareness of these feeds is necessary – and as it is provided by Google and not in-house, may be subject to any problems on Google’s end (downtime, deprecation, etc).

The feeds can be accessed by logging into the administrator account at https://www.google.com/alerts. From there, feeds can be added, deleted, or edited.

Editing Feeds in the Script

If a feed is created, or deleted, in Google Alerts, the script must be edited to reflect that change. This is done by editing the “feed_urls” array in the script (“main.py”, line 32) and adding the feed’s URL to the array (if creating a new script), or removing it (if deleting). In the unlikely event that a feed were to stop working on Google’s service, this is the necessary process to get a copy of the dead feed working.

Google Cloud API

The text and image analysis provided by the application is powered by Google’s Cloud API. As of now, 1,000 image and 5,000 text analysis queries (where 1000 characters = 1 query) are free each month. An additional 3,000 text analyses are allowed by the customers per month (equaling $3.00). The “analysisCap” table in the database tries to keeps track of this and limits API calls accordingly – however, it isn’t an exact science and should be monitored. Actual API calls can be tracked in the Google Cloud console at https://console.cloud.google.com after logging in with the administrator account and selecting the “scotusapp” project. It’s worth mentioning that budget alerts have been set, and at certain thresholds of spending, notification emails will be sent to the administrator account. For more information about the Google Cloud API, visit https://cloud.google.com/apis/.

NewsAPI Backrange Search

By default, NewsAPI searches for articles from the last two days, in case of any downtime. This can be altered by changing the “days=” argument in the timedelta function on line 440 in collectionSources.py, inside the variable “days_ago”.

Topic Sites Search Page Ranges

Certain (but not all) topic sites in the article collector have multiple pages of articles – by default, only the first pages are scraped, but the option is there to scrape a range of these pages by editing the numerical arrays in the functionCalls dict on line 23 in collectionSources.py. The first element is the first page to scrape, the second is the last – the range is inclusive.

Scraping Multiple Images

A feature that is long-requested but has yet to be implemented is the ability to scrape multiple images on article pages. However, the newest version of the application has now provided the backbone to provide it – the database allows for multiple images to belong to the same article, and the new object-oriented nature of the article collector allows us to keep track of images through an array attribute. All that's left is to actually try scraping multiple images from the pages – consider trying this in a future iteration of the application.