



Yioop Documentation v 8.0 (Update In-Progress)

1. [Overview](#)
 - a. [Getting Started](#)
 - b. [Introduction](#)
 - c. [Feature List](#)
2. [Set-up](#)
 - a. [Requirements](#)
 - b. [Installation and Configuration](#)
 - c. [Optional Server and Security Configurations](#)
 - d. [Upgrading Yioop](#)
 - e. [Summary of Files and Folders](#)
3. [Search and User Interface](#)
 - a. [Search Basics](#)
 - b. [Search Page Footer and Search Menu](#)
 - c. [Search Operators](#)
 - d. [Result Formats](#)
 - e. [Mobile Interface](#)
4. [User Accounts and Social Features](#)
 - a. [Registration and Signin](#)
 - b. [Managing Accounts](#)
 - c. [Managing Users, Roles, and Groups](#)
 - d. [Feeds and Wikis](#)
 - e. [Group Feed Chat Bots](#)
5. [Keyword Advertising](#)
 - a. [The Manage Advertisements Activity](#)
 - b. [Bid Mechanics](#)
 - c. [Payment Processing](#)
6. [Crawling and Customizing Results](#)
 - a. [Performing and Managing Crawls](#)
 - b. [Mixing Crawl Indexes](#)
 - c. [Classifying Web Pages](#)
 - d. [Page Indexing and Search Options](#)
 - e. [Web Scrapers](#)
 - f. [Results Editor](#)
 - g. [Search Sources](#)
 - h. [GUI for Managing Machines and Servers](#)
7. [Analytics in Yioop](#)
 - a. [Configuration and Updating](#)
 - b. [Search Analytics](#)
 - c. [Groups Analytics](#)
8. [Building Sites with Yioop](#)
 - a. [Building a Site using Yioop's Wiki System](#)
 - b. [Building a Site using Yioop as Framework](#)
 - c. [Embedding Yioop in an Existing Site](#)
 - d. [Accessing Yioop and getting and OpenSearch RSS, JSON, or JSONP Response](#)



10. [References](#)

Overview

Getting Started

If you have downloaded a prior version of the Yioop software, you may prefer to use one of the following PDF captures of the software documentation:

- [Version 7.1.4: Version 7.1.4 Documentation-PDF](#)
- [Version 6.0.4: Version 6.0.4 Documentation-PDF](#)
- [Version 5.0.4: Version 5.0.4 Documentation-PDF](#)
- [Version 4.0.1: Version 4.0 Documentation-PDF](#)
- [Version 3.20: Version 3.20 Documentation-PDF](#)
- [Version 2.10: Version 2.10 Documentation-PDF](#)

This document serves as a detailed reference for the Yioop search engine. If you want to get started using Yioop now, you probably want to first read the [Installation Guides](#) page and look at the [Yioop Video Tutorials Wiki](#). If you cannot find your particular machine configuration there, you can check the Yioop [Requirements](#) section followed by the more general Installation and Configuration instructions.

[Yioop.com](#), the demo site for Yioop software, allows people to register accounts. Once registered, if you have questions about Yioop and its installation, you can join the [Yioop Software Help](#) group and post your questions there. This group is frequently checked by the creators of Yioop, and you will likely get a quick response.

Having a Yioop account also allows you to experiment with some of the features of Yioop beyond search such as Yioop Groups and Wikis without needing to install the software yourself. The [Search and User Interface](#), [Managing Users, Roles, and Groups](#), and [Feeds and Wikis](#) sections below could serve as a guide to testing the portion of the site general users have access to on Yioop.com.

Activities in Yioop can be found by clicking on the hamburger menu icon . The contents of the menu revealed by clicking on this icon change depending on one of four contexts: Search Web context, Account Admin context, Group Discussion context, and Group Wiki Context. When using Yioop software, if you do not understand a feature, make sure to also check out the integrated help system throughout Yioop. Clicking on a question mark icon will reveal an additional blue column on a page with help information as seen below:

Currently Processing

No Active Crawls

Crawls [\[Query Statistics\]](#)  Row 0 to 10 of 10 Show 50 

New Crawl			
Description:	Start Options ?		
Description:	Timestamp:	Visited/Extracted Urls:	Actions
feed [Statistics]	100 Wed, 31 Dec 1969 16:01:40 -0800	500/500	Closed Set as Index Delete
Genealogy [Statistics]	1548127000 Wed, 31 Dec 1969 16:01:40 -0800	612/4591	Resume Set as Index Delete

New Crawl [\[Edit\]](#) [X]

Enter a name for your crawl and click start to begin a new crawl. Previously completed crawls appear in the table below.

Before you start your crawl be sure to start the queue servers and fetchers to be used for the crawl under **Manage Machines**.

The **Options** link let's you specify what web sites you want to crawl or if you want to do an archive previous crawls or different kinds of data sets.

Most Yioop activities share similar interface features. Namely, the Add Icon  is used to add the main entity controlled by an activity such as a User, Group, Role, Web Crawl, Web Scraper, etc. To open up a search form to search for a particular entity controlled by an activity one clicks on the search icon .

Introduction

The Yioop search engine is designed to allow users to produce indexes of a web-site or a collection of web-sites. The number of pages a Yioop index can handle range from small sites to sites containing tens or hundreds of millions of pages. The largest index so-far created using Yioop is slightly over a billion pages. In contrast, a search-engine like Google



interface to do queries, an rss api, and a function api. It also supports many common features of a search portal such as user discussion group, blogs, wikis, and a news and trends aggregator. In this section we discuss some of the different search engine technologies which exist today, how Yioop fits into this eco-system, and when Yioop might be the right choice for your search engine needs. In the remainder of this document after the introduction, we discuss how to get and install Yioop; the files and folders used in Yioop; the various crawl, search, social portal, and administration facilities in the Yioop; localization in the Yioop system; building a site using the Yioop framework; embedding Yioop in an existing web-site; customizing Yioop; and the Yioop command-line tools. Since the mid-1990s a wide variety of search engine technologies have been explored. Understanding some of this history is useful in understanding Yioop capabilities. In 1994, Web Crawler, one of the earliest still widely-known search engines, only had an index of about 50,000 pages which was stored in an Oracle database. Today, databases are still used to create indexes for small to medium size sites. An example of such a search engine written in PHP is [Sphider](#). Given that a database is being used, one common way to associate a word with a document is to use a table with a columns like word id, document id, score. Even if one is only extracting about a hundred unique words per page, this table's size would need to be in the hundreds of millions for even a million page index. This edges towards the limits of the capabilities of database systems although techniques like table sharding can help to some degree. The Yioop engine uses a database to manage some things like users and roles, but uses its own web archive format and indexing technologies to handle crawl data. This is one of the reasons that Yioop can scale to larger indexes. When a site that is being indexed consists of dynamic pages rather than the largely static page situation considered above, and those dynamic pages get most of their text content from a table column or columns, different search index approaches are often used. Many database management systems like [MySQL/MariaDB](#), support the ability to create full text indexes for text columns. A faster more robust approach is to use a stand-alone full text index server such as [Sphinx](#). However, for these approaches to work the text you are indexing needs to be in a database column or columns, or have an easy to define "XML mapping". Nevertheless, these approaches illustrate another common thread in the development of search systems: Search as an appliance, where you either have a separate search server and access it through either a web-based API or through function calls. Yioop has both a search function API as well as a web API that can return [Open Search RSS](#) results or a JSON variant. These can be used to embed Yioop within your existing site. If you want to create a new search engine site, Yioop provides all the basic features of web search portal. It has its own account management system with the ability to set up groups that have both discussions boards and wikis with various levels of access control. The built in Public group's wiki together with the GUI configure page can be used to completely customize the look and feel of Yioop. Third party display ads can also be added through the GUI interface. If you want further customization, Yioop offers a web-based, model-view-adapter (a variation on model-view-controller) framework with a web-interface for localization.

By 1997 commercial sites like Inktomi and AltaVista already had tens or hundreds of millions of pages in their indexes [[P1994](#)] [[P1997a](#)] [[P1997b](#)]. Google [[BP1998](#)] circa 1998 in comparison had an index of about 25 million pages. These systems used many machines each working on parts of the search engine problem. On each machine there would, in addition, be several search related processes, and for crawling, hundreds of simultaneous threads would be active to manage open connections to remote machines. Without threading, downloading millions of pages would be very slow. Yioop is written in [PHP](#). This language is the 'P' in the very popular [LAMP](#) web platform. This is one of the reasons PHP was chosen as the language of Yioop. Unfortunately, PHP does not have built-in threads. However, the PHP language does have a multi-curl library (implemented in C) which uses threading to support many simultaneous page downloads. This is what Yioop uses. Like these early systems Yioop also supports the ability to distribute the task of



http.

There are several aspects of a search engine besides downloading web pages that benefit from a distributed computational model. One of the reasons Google was able to produce high quality results was that it was able to accurately rank the importance of web pages. The computation of this page rank involves repeatedly applying Google's normalized variant of the web adjacency matrix to an initial guess of the page ranks. This problem naturally decomposes into rounds. Within a round the Google matrix is applied to the current page ranks estimates of a set of sites. This operation is reasonably easy to distribute to many machines. Computing how relevant a word is to a document is another task that benefits from multi-round, distributed computation. When a document is processed by indexers on multiple machines, words are extracted and a stemming algorithm such as [[P1980](#)] or a character n-gramming technique might be employed (a stemmer would extract the word jump from words such as jumps, jumping, etc; converting jumping to 3-grams would make terms of length 3, i.e., jum, ump, mpi, pin, ing). For some languages like Chinese, where spaces between words are not always used, a segmenting algorithm like reverse maximal match might be used. Next a statistic such as BM25F [[ZCTSR2004](#)] (or at least the non-query time part of it) is computed to determine the importance of that word in that document compared to that word amongst all other documents. To do this calculation one needs to compute global statistics concerning all documents seen, such as their average-length, how often a term appears in a document, etc. If the crawling is distributed it might take one or more merge rounds to compute these statistics based on partial computations on many machines. Hence, each of these computations benefit from allowing distributed computation to be multi-round. Infrastructure such as the Google File System [[GGL2003](#)], the MapReduce model [[DG2004](#)], and the Sawzall language [[PDGQ2006](#)] were built to make these multi-round distributed computation tasks easier. In the open source community, the [Hadoop Distributed File System](#), [Hadoop MapReduce](#), and [Pig](#) play an analogous role [[W2009](#)]. More recently, a theoretical framework for what algorithms can be carried out as rounds of map inputs to sequence of key value pairs, shuffle pairs with same keys to the same nodes, reduce key-value pairs at each node by some computation has begun to be developed [[KSV2010](#)]. This framework shows the map reduce model is capable of solving quite general cloud computing problems -- more than is needed just to deploy a search engine.

Infrastructure such as this is not trivial for a small-scale business or individual to deploy. On the other hand, most small businesses and homes do have available several machines not all of whose computational abilities are being fully exploited. Also, it is relatively cheap to rent multiple machines in a cloud service. So the capability to do distributed crawling and indexing in this setting exists. Further high-speed internet for homes and small businesses is steadily getting better. Since the original Google paper, techniques to rank pages have been simplified [[APC2003](#)]. It is also possible to approximate some of the global statistics needed in BM25F using suitably large samples. More details on the exact ranking mechanisms used by Yioop and be found on the [Yioop Ranking Mechanisms](#) page. Yioop tries to exploit these advances to use a simplified distributed model which might be easier to deploy in a smaller setting. Each node in a Yioop system is assumed to have a web server running. One of the Yioop nodes web app's is configured to act as a coordinator for crawls. It is called the **name server** . In addition to the name server, one might have several processes called **queue servers** that perform scheduling and indexing jobs, as well as **fetcher** processes which are responsible for downloading pages and the page processing such as stemming, char-gramming and segmenting mentioned above. Through the name server's web app, users can send messages to the queue servers and fetchers. This interface writes message files that queue servers periodically looks for. Fetcher processes periodically ping the name server to find the name of the current crawl as well as a list of queue servers. Fetcher programs then periodically make requests in a round-robin fashion to the queue servers for messages and schedules. A schedule is data to process and a message has control information about what kind of processing should be done. A



the responsible queue server's web server. The data is then written to a set of received files. The queue server as part of its loop looks for received files and merges their results into the index so far. So the model is in a sense one round: URLs are sent to the fetchers, summaries of downloaded pages are sent back to the queue servers and merged into their indexes. As soon as the crawl is over one can do text search on the crawl. Deploying this computation model is relatively simple: The web server software needs to be installed on each machine, the Yioop software (which has the the fetcher, queue server, and web app components) is copied to the desired location under the web server's document folder, each instance of Yioop is configured to know who the name server is, and finally, the fetcher programs and queue server programs are started.

As an example of how this scales, 2010 Mac Mini running a queue server program can schedule and index about 100,000 pages/hour. This corresponds to the work of about 7 fetcher processes (which may be on different machines -- roughly, you want 1GB and 1core/fetcher). The checks by fetchers on the name server are lightweight, so adding another machine with a queue server and the corresponding additional fetchers allows one to effectively double this speed. This also has the benefit of speeding up query processing as when a query comes in, it gets split into queries for each of the queue server's web apps, but the query only "looks" slightly more than half as far into the posting list as would occur in a single queue server setting. To further increase query throughput, the number queries that can be handled at a given time, Yioop installations can also be configured as "mirrors" which keep an exact copy of the data stored in the site being mirrored. When a query request comes into a Yioop node, either it or any of its mirrors might handle it. Since a multi-million page crawl involves both downloading from the web rapidly over several days, Yioop supports the ability to dynamically change its crawl parameters as a crawl is going on. This allows a user on request from a web admin to disallow Yioop from continuing to crawl a site or to restrict the number of urls/hours crawled from a site without having to stop the overall crawl. One can also through a web interface inject new seed sites, if you want, while the crawl is occurring. This can help if someone suggests to you a site that might otherwise not be found by Yioop given its original list of seed sites. Crawling at high-speed can cause a website to become congested and unresponsive. As of Version 0.84, if Yioop detects a site is becoming congested it can automatically slow down the crawling of that site. Finally, crawling at high-speed can cause your domain name server (the server that maps [www.yioop.com](#) to 173.13.143.74) to become slow. To reduce the effect of this Yioop supports domain name caching.

Despite its simpler one-round model, Yioop does a number of things to improve the quality of its search results. While indexing, Yioop can make use Lasso regression classifiers [[GLM2007](#)] using data from earlier crawls to help label and/or rank documents in the active crawl. Yioop also takes advantage of the link structure that might exist between documents in a one-round way: For each link extracted from a page, Yioop creates a micropage which it adds to its index. This includes relevancy calculations for each word in the link as well as an [[APC2003](#)]-based ranking of how important the link was. Yioop supports a number of iterators which can be thought of as implementing a stripped-down relational algebra geared towards word-document indexes (this is much the same idea as Pig). One of these operators allows one to make results from unions of stored crawls. This allows one to do many smaller topic specific crawls and combine them with your own weighting scheme into a larger crawl. A second useful operator allows you to display a certain number of results from a given subquery, then go on to display results from other subqueries. This allows you to make a crawl presentation like: the first result should come from the open crawl results, the second result from Wikipedia results, the next result should be an image, and any remaining results should come from the open search results. Yioop comes with a GUI facility to make the creation of these crawl mixes easy. To speed up query processing for these crawl mixes one can also create materialized versions of crawl mix results, which makes a separate index of crawl mix results. Another useful operator Yioop supports allows one to perform groupings of document results. In the search results displayed, grouping by



low ranked link with the word "stupid" to a given site would tend not to show up early in the results for the word "stupid". Grouping also is used to handle deduplication: It might be the case that the pages of many different URLs have essentially the same content. Yioop creates a hash of the web page content of each downloaded url. Amongst urls with the same hash only the one that is linked to the most will be returned after grouping. Finally, if a user wants to do more sophisticated post-processing such as clustering or computing page rank, Yioop supports a straightforward architecture for indexing plugins.

There are several open source crawlers which can scale to crawls in the millions to hundred of millions of pages. Most of these are written in Java, C, C++, C#, not PHP. Three important ones are [Nutch](#)/[Lucene](#)/[Solr](#) [[KC2004](#)], [YaCy](#), and [Heritrix](#) [[MKSR2004](#)]. Nutch is the original application for which the Hadoop infrastructure described above was developed. Nutch is a crawler, Lucene is for indexing, and Solr is a search engine front end. The YaCy project uses an interesting distributed hash table peer-to-peer approach to crawling, indexing, and search. Heritrix is a web crawler developed at the [Internet Archive](#). It was designed to do archival quality crawls of the web. Its ARC file format inspired the use of WebArchive objects in Yioop. WebArchives are Yioop's container file format for storing web pages, web summary data, url lists, and other kinds of data used by Yioop. A WebArchive is essentially a linked-list of compressed, serialized PHP objects, the last element in this list containing a header object with information like version number and a total count of objects stored. The compression format can be chosen to suit the kind of objects being stored. The header can be used to store auxiliary data structures into the list if desired. One nice aspect of serialized PHP objects versus serialized Java Objects is that they are humanly readable text strings. The main purpose of Web Archives is to allow one to store many small files compressed as one big file. They also make data from web crawls very portable, making them easy to copy from one location to another. Like Nutch and Heritrix, Yioop also has a command-line tool for quickly looking at the contents of such archive objects.

The [ARC format](#) is one example of an archival file format for web data. Besides at the Internet Archive, ARC and its successor [WARC format](#) are often used by TREC conferences to store test data sets such as [GOV2](#) and the [ClueWeb 2009](#) / [ClueWeb 2012](#) Datasets. In addition, it was used by grub.org (hopefully, only on a temporary hiatus), a distributed, open-source, search engine project in C#. Another important format for archiving web pages is the XML format used by Wikipedia for archiving MediaWiki wikis. [Wikipedia](#) offers [creative common-licensed downloads](#) of their site in this format. [Curlie.org](#), formerly the Open Directory Project, makes available its [ODP data sets](#) in an RDF-like format licensed using the Open Directory License. Thus, we see that there are many large scale useful data sets that can be easily licensed. Raw data dumps do not contain indexes of the data though. This makes sense because indexing technology is constantly improving and it is always possible to re-index old data. Yioop supports importing and indexing data from ARC, WARC, database queries results, MediaWiki XML dumps, and Open Directory RDF. Yioop further has a generic text importer which can be used to index log records, mail, Usenet posts, etc. Yioop also supports re-indexing of old Yioop data files created after version 0.66, and indexing crawl mixes. This means using Yioop you can have searchable access to many data sets as well as have the ability to maintain your data going forward. When displaying caches of web pages in Yioop, the interface further supports the ability to display a history of all cached copies of that page, in a similar fashion to Internet Archives interface.

Another important aspect of creating a modern search engine is the ability to display in an appropriate way various media sources. Yioop comes with built-in subsearch abilities for images, where results are displayed as image strips; video, where thumbnails for video are shown; news, where news items can be grouped together by category (Headlines, Sports, Entertainment) and a configurable set of news/twitter feeds can be set to be updated on an hourly basis; and trends, where statistics about keywords in news items or on web pages change in time are presented.



The feature changes between different versions of Yioop can be found on the [Changelog](#) page. Here is a summary of the main features of the current version of Yioop:

- **General**

- Yioop is an open-source, distributed crawler and search engine written in PHP.
- Yioop has its own built-in web server so can be deployed with or without a web server such as Apache. Yioop also has been tested to work when Apache is configured to use HTTP/2.0.
- Crawling, indexing, and serving search results can be done on a single machine or distributed across several machines.
- The fetcher/queue server processes on several machines can be managed through the web interface of a main Yioop instance.
- Yioop installations can be created with a variety of topologies: one queue server and many fetchers or several queue servers and many fetchers.
- Using web archives, crawls can be mirrored amongst several machines to speed-up serving search results. This can be further sped-up by using filecache.
- Yioop can be used to create web sites via its own built-in wiki system. For more complicated customizations, Yioop's model-view-adapter framework is designed to be easily extendible. This framework also comes with a GUI which makes it easy to localize strings and static pages.
- Yioop can be used as a package in other project using [Composer](#) .
- Yioop search result and feed pages can be configured to display banner or skyscraper ads through an Site Admin GUI (if desired).
- Yioop search result and feed pages can also be configured to use Yioop's built-in keyword advertising system.
- Yioop has been optimized to work well with smart phone web browsers and with tablet devices.
- Yioop has a built in analytics system which when running can be used to track search queries, wiki page views, and discussion board views.
- To help ensure privacy when search and discussion board aggregate data is displayed Yioop can make use of a differential privacy subsystem.
- Yioop can be configured using its web interface including image, audio, and video serving.

- **Social and User Interface**

- Yioop can be configured to allow or not to allow users to register for accounts.
- If allowed, user accounts can create discussion groups, blogs, and wikis.
- On a per group basis, monetization in Yioop is also supported by charging a credit fee to join groups.
- Blogs and wiki support attaching images, videos, and files, have a facility for displaying QR codes, and also support including math using LaTeX or AsciiMathML.
- Discussion boards support [Chat Bots](#) and Yioop has an API for integrating Chat Bot Users into groups.
- Yioop comes with two built in groups: Help and Public. Help's wiki pages allow one to customize the integrated help throughout the Yioop system. The Public Groups discussion can be used as a site blog, its wiki page can be used to customize the look-and-feel of the overall Yioop site without having to do programming.
- Data from non-Yioop discussion boards, if presented as RSS, can be imported into a Yioop discussion group.
- Wiki pages support different types such as standard wiki page, wiki page with discussions, media gallery, page alias, share wall, slide presentation, and URL Shortener. Media galleries can be used to present lists of videos together with whether they have been watched, and if so, how far -- much like Kodi.



- Video on wiki pages and in discussion posts is served using HTML5 pseudo-streaming so users can scrub through video files. For uploaded video files below a configurable size limit, videos are automatically converted to web friendly mp4 format, provided the distributed version of the media updater is in use.
 - If FFmpeg is configured thumbnails of videos will show animated snippets.
 - Yioop detects automatically if a video has associated with it a [VTT](#) captioning or subtitling file and can serve videos with captions and subtitles.
 - PDF and Epub files can be displayed automatically on gallery pages, allowing simple e-reading which remembers where a user was last reading.
 - Wiki pages can be configured to have auto-generated tables of contents, to make use of common headers and footers, and to output meta tags for SEO purposes.
 - Users can share their own mixes of crawls that exist in the Yioop system.
 - If user accounts are enabled, Yioop has a search tools page on which people can suggest urls to crawl.
 - Yioop has three different captcha'ing mechanisms that can be used in account registration and for suggest urls: a standard graphics-based captcha, a text-based captcha, and a hash-cash-like catpha.
 - Password authentication can be configured to either use a standard password hash based system, or make use of Fiat Shamir zero-knowledge authentication.
- **Search**
 - Yioop supports subsearches geared towards presenting certain kinds of media such as images, video, and news. The list of news sites and how videos are scraped can be configured through the GUI. Yioop has a media updater process which can be used to automatically update news and general media feeds hourly. It can also be used to download podcasts to wiki pages.
 - Yioop has a trending page for what words were most popular in the news feeds at different time scales.
 - New and media feeds can either be RSS feeds, Atom feeds, JSON feeds, or can be scraped from an HTML page using XPath queries or can be scraped using a regular expression scaper. What image is used for a news feed item can also be configured using XPath queries.
 - Several hourly processes such as news updates, podcast updates, video conversion, and mail delivery can be configured to be distributed across several machines automatically.
 - Yioop determines search results using a number of iterators which can be combined like a simplified relational algebra.
 - Yioop can be configured to display word suggestions as a user types a query. It can also suggest spell corrections for mis-typed queries. This feature can be localized.
 - Yioop has a built-in triplet extraction system for English which allows for a limited form of Question Answering within Yioop for English.
 - Yioop supports language detection of queries and web pages and safe search of web pages.
 - Yioop supports the ability to filter out urls from search results after a crawl has been performed. It also has the ability to edit summary information that will be displayed for urls.
 - A given Yioop installation might have several saved crawls and it is very quick to switch between any of them and immediately start doing text searches.
 - Besides the standard output of a web page with ten links, Yioop can output query results in Open Search RSS format, a JSON variant of this format, and also to query Yioop data via a function api.
 - **Indexing**
 - Yioop is capable of indexing small sites to sites or collections of sites containing



- Yioop has a web interface which makes it easy to combine results from several crawl indexes to create unique result presentations. These combinations can be done in a conditional manner using "if:" meta words.
- Yioop supports the indexing of many different filetypes including: HTML, Atom, BMP, DOC, DOCX ePub, GIF, JPG, PDF, PPT, PPTX, PNG, RSS, RTF, sitemaps, SVG, XLSX, and XML. It has a web interface for controlling which amongst these filetypes (or all of them) you want to index. It supports also attempting to extract information from unknown filetypes.
- Yioop supports extracting data from zipped formats like DOCX even if it only did a partial download of the file.
- Yioop has a simple page rule language for controlling what content should be extracted from a page or record.
- Users can define rules for web scraping content from particular kinds of web sites such as Wordpress, Drupal, and other CMS systems and users can define rules to scrape different kinds of meta information such as Open Graph tags.
- Yioop has four different kinds of text summarizers which can be used to further affect what words are index: a basic tag-based summarizer, a graph-based summarizer, and a centroid and weighted-centroid algorithm summarizer. It can generate word clouds from summarized documents.
- Indexing occurs as crawling happens, so when a crawl is stopped, it is ready to be used to handle search queries immediately.
- Yioop Indexes can be used to create classifiers which then can be used in labeling and ranking future indexes.
- Yioop comes with stemmers for English, Arabic, Dutch, French, German, Hindi, Italian, Persian, Portuguese, Russian, and Spanish, and a word segmenter for Chinese. It uses char-gramming for other languages. Yioop has a simple architecture for adding stemmers for other languages.
- Yioop uses a web archive file format which makes it easy to copy crawl results amongst different machines. It has a command-line tool for inspecting these archives if they need to be examined in a non-web setting. It also supports command-line search querying of these archives.
- Yioop supports an indexing plugin architecture to make it possible to write one's own indexing modules that do further post-processing.
- **Web and Archive Crawling**
 - Yioop supports open web crawls, but through its web interface one can configure it also to crawl only specific sites, domains, or collections of sites and domains.
 - Yioop supports crawling a site up to a certain depth starting from seed nodes.
 - Yioop supports general repeating crawls. These crawls have a repeat frequency and two indexes: one for searching for crawling and Yioop automatically switches between the two every repeat period.
 - Yioop supports multiple simultaneous crawls by assigning machines to channels and then scheduling crawls to those channels.
 - Yioop uses multi-curl to support many simultaneous downloads of pages.
 - Yioop obeys robots.txt files by default including Google and Bing extensions such as the Crawl-delay and Sitemap directives as well as * and \$ in allow and disallow. It further supports the robots meta tag directives NONE, NOINDEX, NOFOLLOW, NOARCHIVE, and NOSNIPPET and the link tag directive rel="canonical". It also supports anchor tags with rel="nofollow" attributes. It also supports X-Robots-Tag HTTP headers. Finally, it tries to detect if a robots.txt became a redirect due to congestion.
 - Yioop can be configured to relax how it obeys robots.txt to allow it to always download landing page, or to completely ignore the robots.txt if one wants.
 - Yioop sends referer headers when crawling pages which when not present are



- more subject specific indexes.
- Yioop has its own DNS caching mechanism and it adjusts the number of simultaneous downloads it does in one go based on the number of lookups it will need to do.
 - Yioop can crawl over HTTP, HTTPS, and Gopher protocols. Yioop is configured to try to do HTTP requests over HTTP/2.0 before falling back to HTTP/1.1 or HTTP/1.0.
 - Yioop supports crawling TOR networks (.onion urls).
 - Yioop supports crawling through a list of proxy servers.
 - Yioop supports crawling Git Repositories and can index Java and Python code.
 - Yioop supports crawl quotas for web sites. I.e., one can control the number of urls/hour downloaded from a site.
 - Yioop can detect website congestion and slow down crawling a site that it detects as congested.
 - Yioop supports dynamically changing the allowed and disallowed sites while a crawl is in progress. Yioop also supports dynamically injecting new seeds site via a web interface into the active crawl.
 - Yioop has a web form that allows a user to control the recrawl frequency for a page during a crawl.
 - Yioop keeps track of ETag: and Expires: HTTP headers to avoid downloading content it already has in its index.
 - Yioop supports importing data from ARC, WARC, database queries, MediaWiki XML, and ODP RDF files. It has generic importing facility to import text records such as access log, mail log, usenet posts, etc., which are either not compressed, or compressed using gzip or bzip2. It also supports re-indexing of data from WebArchives.

[Return to table of contents](#) .

Set-up Requirements

Yioop can be configured to run using its internal web-server or using an external web server. Run as its own web server, Yioop requires: (1) PHP 7.1 or better, (2) Curl libraries enabled in PHP, (3) sqlite support enabled in PHP, (4) multi-byte string support enabled in PHP, (5) xml support enabled in PHP, (6) GD graphic libraries enabled in PHP, (7) openssl support enabled in PHP. Most of these features are enabled in PHP by default, but you should still check your configuration. In a third party we server setting, the Yioop search engine requires in addition: (1) a web server, (2) Rewrites enabled. To be a little more specific Yioop has been tested with Apache 2.2, Apache 2.4 and I've been told Version 0.82 or newer works with lighttpd. It should work with other webservers, although it might take some finessing. If you are using MacOS Snow Leopard or newer, the version of Apache 2 and PHP that comes with it suffice. For Windows, Mac, and Linux, another easy way to get the required software is to download a Apache/PHP/MySQL suite such as [XAMPP](#).

On any platform, if you decide to run Yioop under Apache, make sure that the Apache mod_rewrite module is enabled and that .htaccess files work for the directory in question (often this is the case by default with packages such as Xampp, but please check the Yioop install guide for the platform in question in order to see if anything is required). It is possible to get Yioop to work without mod_rewrite. To do so, you can use the src directory location as the url for Yioop, however, URLs in this scenario used by your installation will look uglier. On Windows machines, find the the php.ini file under the php folder in your Xampp folder and change the line:

```
;extension=php_curl.dll
```

to

```
post_max_size = 32M
```

Yioop will work with the `post_max_size` set to as little as two megabytes bytes, but will be faster with the larger post capacity. If you intend to make use of Yioop Discussion Groups and Wikis and their ability to upload documents, you might want to consider also adjusting the value of the variable `upload_max_filesize`. This value should be set to at most what you set `post_max_size` to.

If you are using WAMP, similar changes as with XAMPP must be made, but be aware that WAMP has two `php.ini` files and both of these must be changed.

If you are using the Ubuntu-variant of Linux, the following lines would get the software you need to run Yioop without a web server: `sudo apt install curl`

```
sudo apt install php-cli
sudo apt install php-mbstring
sudo apt install php-sqlite
sudo apt install php-curl
sudo apt install php-gd
sudo apt install php-xml
sudo apt install php-bcmath
```

With a web server, you should also install:

```
sudo apt install apache2
sudo apt install php
sudo apt install libapache2-mod-php
sudo a2enmod php
sudo a2enmod rewrite
```

If you decide to use a different database than `sqlite`, you need the appropriate `php` driver. For example, for `mysql`:

```
sudo apt install php-mysql
```

For both Mac and Linux, you might want to alter the `post_max_size` variable in your `php.ini` file as in the Windows case above.

In addition to the minimum installation requirements above, if you want to use the [Manage Machines](#) feature in Yioop, you may need to do some additional configuration. Namely, some sites disable the PHP `popen`, `pclose`, and `exec` functions, and so you might have to edit your `php.ini` file to enable these functions if you want this activity to work. The Manage Machines activity allows you through a web interface to start/stop and look at the log files for each of the QueueServer's, and Fetchers that you want Yioop to manage. It also allows you to start and stop the Media Updater process/processes. If it is not configured then these task would need to be done via the command line. **Also, if you do not use the Manage Machine interface your Yioop site can make use of only one QueueServer.** As a final step, after installing the necessary software, **make sure to start/restart your web server and verify that it is running.**

Memory Requirements

Yioop sets for its processes certain upper bounds on the amounts of memory they can use. These are partially calculated based on the amount of available memory your computer has. For machines with at least 8GB of RAM, by default `src/executables/QueueServer.php`'s limit is set to 4GB, `src/executables/Fetcher.php`'s limit is set to 2GB and for `index.php` the limit is 1GB. For a machine with 4GB of RAM the values are half of these. These values in turn affect the size of crawling objects such as the in-memory priority queue, etc. Each processes memory value is set near the tops of its files with a line like:

```
ini_set("memory_limit", C\FETCHER_MEMORY_LIMIT);
```

where the constant `FETCHER_MEMORY_LIMIT` is computed in `src/configs/Config.php`.

For the `index.php` file, you may need to set the limit as well in your `php.ini` file for the instance of PHP used by your web server. If the value is too low for the `index.php` Web app, you might see messages in the Fetcher logs that begin with: "Trouble sending to the scheduler at url..."

Often in a VM setting these requirements are somewhat steep. It is possible to get Yioop to work in environments like EC2 (be aware this might violate your service agreement). To



src/settings/setting.php or in the file src/settings/localconfig.php . Experimenting with these values you should be able to trade-off memory requirements for speed in a more fine-grained manner.

LocalConfig.php File

In addition to the settings mentioned above under memory requirements, several other default settings can be overridden in a LocalConfig.php file. If you look at the src/configs /Config.php file any constant defined by a function call **nsconddefine** can be overridden. We list a couple useful ones here:

ALLOW_FREE_ROOT_CREDIT_PURCHASE

allows the root account to purchase add credits for free.

DIRECT_ADD_SUGGEST

makes it so user suggested urls are directly added to the current crawl, rather than added only when a user clicks a link on the Manage Crawls page.

FFMPEG

used to set the path to FFMPEG (if it is installed) that can used to convert uploaded videos to groups to mp4. It is also used to make animated GIFs for thumbnails of videos.

MAX_QUERY_CACHE_TIME

controls how long a query is cached before it is recomputed. A value of -1 keeps it in the cache until the cache is full.

PHANTOM_JS

sets path to Phantom JS headless Chrome browser (if it is installed -- note: the Phantomjs project is no longer maintained) that is used if one wants to run the Javascript unit tests for the site.

QRENCODE

used to set the path to qrencode (if it is installed) that can used to add QR codes to wiki pages.

ROOT_USERNAME

allows one to change the root username to something other than root

SERVER_CONTEXT

used to set up the way the built-in web server works in Yioop. The value of this variable is an array of configuration parameters, for example:

```
['SERVER_ADMIN' => 'bob@builder.org',  
'SERVER_NAME' => 'Yioop',  
'SERVER_SOFTWARE' => 'YIOOP_SERVER',  
'ssl' => [  
  'local_cert' => 'your_cert.crt',  
  'cafile' => 'certificate_authority.crt',  
  'capath' => '/etc/ssl/certs',  
  'local_pk' => '/etc/ssl/private/path_to_your_private.key',  
  'allow_self_signed' => false,  
  'verify_peer' => false,  
],  
'USER' => 'cpollett'] # here user is the *nix user to run under.
```

TESSERACT

path to tesseract command line utility for recognizing text from images.

If installed, Yioop will recognize text in images and PDFs.

[Return to table of contents](#) .

Installation and Configuration

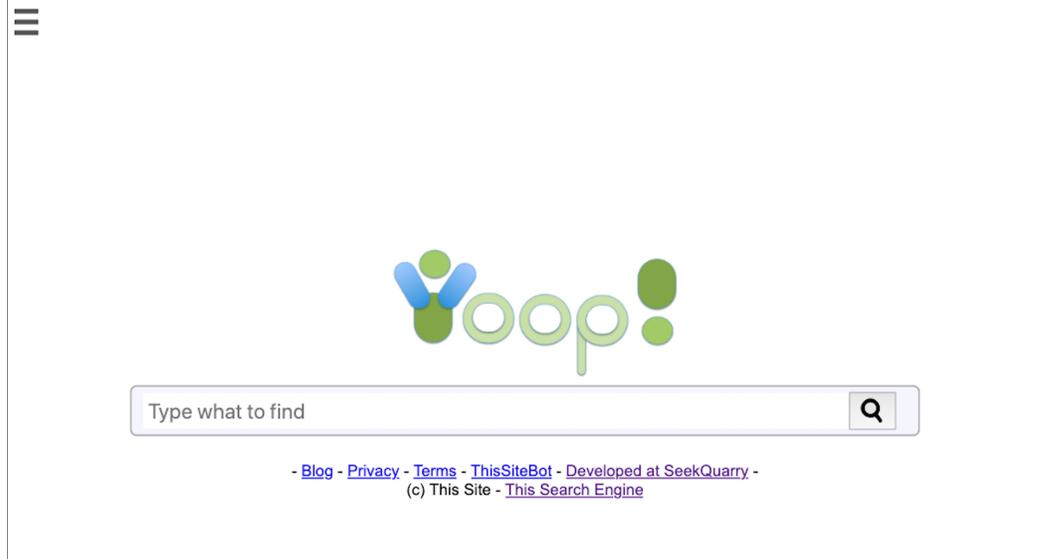
The Yioop application can be obtained using [the download page at seekquarry.com](#) . After downloading and unzipping it, move the Yioop search engine into the folder where you'd like to keep it. If you are running Yioop with an external web server such as Apache, this should be under your web server's document root. If you are not using an external web server, on Linux or Mac open a terminal shell, on a Windows machine open the command

php index.php

or

php index.php port_number_to_run_on

Open a browser, and go to the page `http://localhost:8080/` . If you are running under an external web server, go to the url corresponding to where you placed Yioop under document root. You should see the default Yioop landing page:



If you don't see this page, check that you have installed and configured all the prerequisite software in the [Requirements section](#) . The most common sources of problems are that `mod_rewrite` package is not loaded for Apache, the `.htaccess` file of Yioop is not being processed because Apache is configured so as not to allow overrides on that directory, or the webserver doesn't have write permissions on `YIOOP_DIR/src/configs/Configs.php` and the `YIOOP_DIR/work_directory folder` . If you have gotten the above screen, then congratulations, Yioop is installed! In this section, we will give a first pass at how to customize some general aspects of your site. These will include configuring your site for development versus production, describing your crawler to the websites you will crawl, customizing the icons, color scheme, and timezone settings of your site. The next section, describes more advanced configurations such as user registration and advertisement serving.

Configurations for Development, Production, Testing, and Crawling

Click on the Sign In link to log on to the administrative panel for Yioop. The default username is `root` with an empty password. Under the **System Settings** activity group on the side of the screen, click on the **Configure Activity** . This activity page looks like:



/Applications/MAMP/htdocs/git/yioop2/work_direct ?

Component Check

Check Passed.

Profile Settings

العربية
বাংলা
Default Language: Deutsch
English
Español
عربي

Debug Display ?

Error Info Query Info Test Info

Search Access ?

Web RSS API

Crawl Robot Set-up ?

Crawl Robot Name:

Robot Instance:

Robot Description

```
title=Bot
description=Describes the web crawler used with this
web site
END_HEAD_VARS
==My Web Crawler==
Please Describe Your Robot
```

[- Blog](#) - [- Privacy](#) - [- Terms](#) - [- ThisSiteBot](#) - [- Developed at SeekQuarry](#) -
(c) This Site - [This Search Engine](#)

The Search Engine Work Directory form let's you specify the name of a folder where all of the data specific to your copy of Yioop should be stored. This defaults to YIOOP_DIR/work_directory, and you probably don't want to change this, but you can customize its locations to where you want with this form. The web server needs permissions to be able to write this directory for Yioop to work. Notice under the text field there is a heading "Component Check" and it says "Checks Passed". Yioop does an automatic check to see if it can find all the common built-in classes and functions it needs to run, making sure they have not been disabled. If says something other than "Checks passed", you will probably need to edit your php.ini file to get Yioop to be able to do crawls. In the above Configure Screen image, there is a Profile Settings form beneath the Search Engine Work Directory form. The Profile Settings form allows you to configure the debug, search access, database, queue server, and robot settings. The **Debug Display** fieldset has three checkboxes: Error Info, Query Info, and Test Info. Checking Error Info will mean that when the Yioop web app runs, any PHP Errors,



be displayed at the bottom of each web page. The last checkbox, Test me, says whether or not to display automated tests of some of the systems library classes, the link next to this goes to a page where you can run the tests. None of these debug settings should be checked in a production environment.

The **Search Access** fieldset has three checkboxes: Web, RSS, and API. These control whether a user can use the web interface to get query results, whether RSS responses to queries are permitted, or whether or not the function based search API is available. Using the Web Search interface and formatting a query url to get an RSS response are describe in the Yioop [Search and User Interface](#) section. The Yioop Search Function API is described in the section [Embedding Yioop](#) , you can also look in the examples folder at the file SearchApi.php to see an example of how to use it. **If you intend to use Yioop in a configuration with multiple queue servers (not fetchers), then the RSS checkbox needs to be checked.**

The **Crawl Robot Set-up** fieldset is used to provide websites that you crawl with information about who is crawling them.

- The field **Crawl Robot Name** is used to say part of the USER-AGENT. It has the format:

```
Mozilla/5.0 (compatible; NAME_FROM_THIS_FIELD; YOUR_SITES_URL/bot)
```

The value set will be common for all fetcher traffic from the same queue server on site when downloading webpages. If you are doing crawls using multiple queue servers you should give the same value to each queue server. The value of YOUR_SITES_URL comes from the Server Settings - Name Server URL field.

- The **Robot Instance** field is used for web communication internal to a single yioop instance to help identify which queue server or fetcher under that queue server was involved. This string should be unique for each queue server in your Yioop set-up. The value of this string is written when logging requests between fetchers and queue servers and can be helpful in debugging.
- The **Robot Description** field is used to specify the Public bot wiki page. This page can also be accessed and edited under Manage Groups by clicking on the wiki link for the Public group and then editing its Bot page. This wiki page is what's display when someone goes to the URL:

```
YOUR_SITES_URL/bot
```

The point of this page is to give web owners both contact info for your bot as well as a description of how your bot crawls web sites.

Changing the Global Appearance of Yioop

To change the global appearance of Yioop, under the **System Settings** activity group on the side of the screen, click on the **Appearance** . This activity page looks like:

[\[Adjust Search Page Components\]](#)

Background Color:

Background Image: 

Drag above or ...
[choose](#) Upload

Foreground Color:

Top Bar Color:

Side Bar Color:

Large Logo: 

Drag above or ...
[choose](#) Upload

Medium Logo: 

Drag above or ...
[choose](#) Upload

Small Logo: 

Drag above or ...
[choose](#) Upload

Favicon: 

Drag above or ...
[choose](#) Upload

Search Toolbar 

Drag above or ... [choose](#) Upload

Auxiliary Style Directives

[\[Reset Customizations\]](#)

Save

The **Use Wiki Public Main Page as Landing Page** checkbox lets you set the main page of the Public wiki to be the landing page of the whole Yioop site rather than the default centered search box landing page. Several of the text fields in Site Customizations control various colors used in drawing the Yioop interface. These include **Background Color** , **Foreground Color** , **Top Bar Color** , **Side Bar Color** . The values of these fields can be any legitimate style-sheet color such as a # followed by an red, green, blue value (between 0-9 A-F), or a color word such as: yellow, cyan, etc. If you would like to use a background image, you can either use the picker link or drag and drop one into the rounded square next



A Search Feature is a short file that can be used to add your search engine to the search bar of a browser. You can drag such a file into the gray area next to this label and click save to set this for your site. The link to install the search bar is visible on the Settings page.

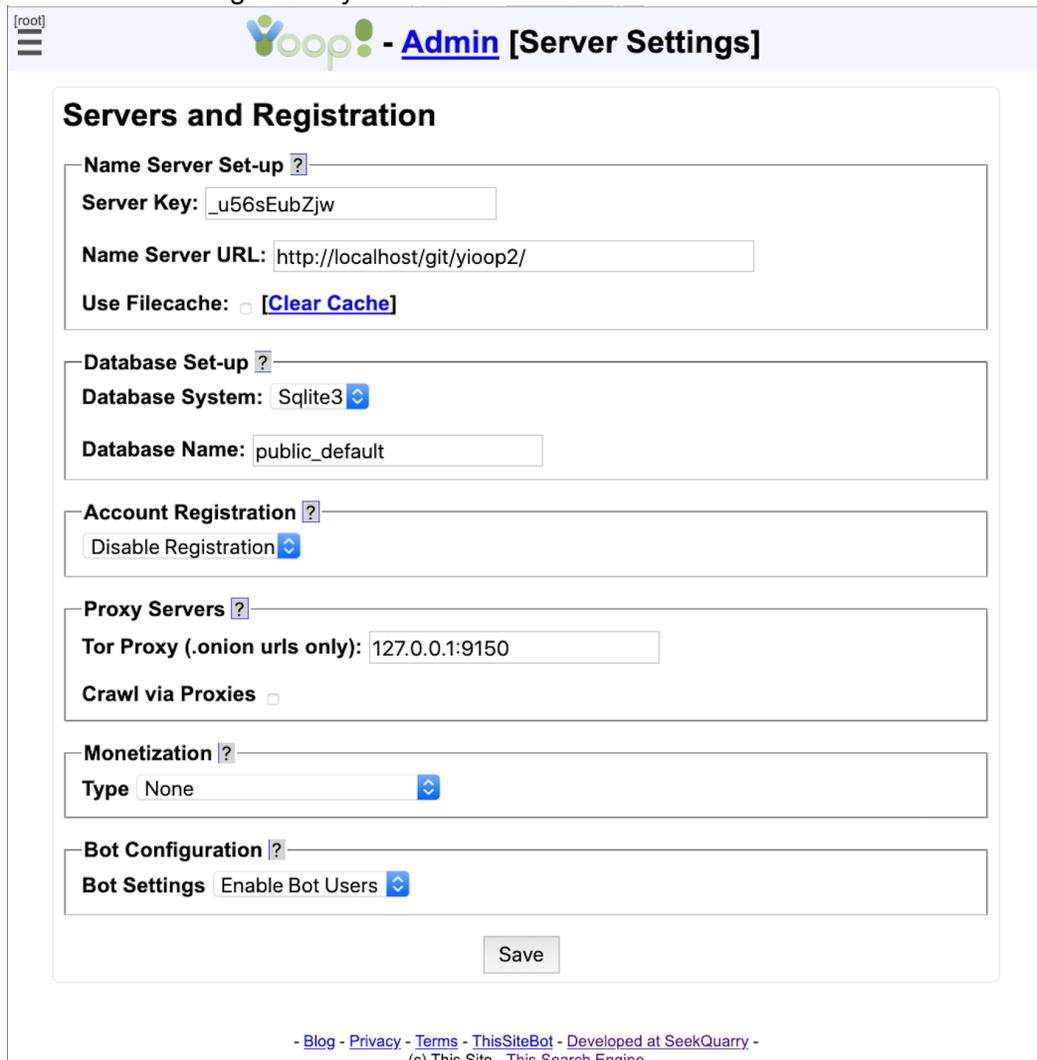
There is also a link tag on every page of the Yioop site that allows a browser to auto-discover this as well. As a starting point, one can try tweaking the default Yioop search bar, yioopbar.xml, in the base folder of Yioop.

Finally, if one knows cascading stylesheets (CSS) and wants greater control of the the look and feel of the site, then one can enter standard stylesheet command in the **Auxiliary Style Directives** textarea.

Optional Server and Security Configurations

The configuration and appearance activities just described suffices to set up Yioop for a single server crawl and do simple customization on how your Yioop site looks. If that is all that you are interested in, you may want to skip ahead to the section on the [Yioop Search Interface](#) to learn about the different search features available in Yioop or you may want to skip ahead to [Performing and Managing Crawls](#) to learn about how to perform a crawl. In this section, we describe the Server Settings and Security activities which might be useful in a multi-machine, multi-user setting and which might also be useful for crawling hidden websites or crawling through proxies. We also describe the form used to configure advertisements in Yioop.

The Server Settings activity looks like:



The screenshot shows the 'Server Settings' page in the Yioop Admin interface. The page title is 'Yioop! - Admin [Server Settings]'. The main heading is 'Servers and Registration'. The settings are organized into several sections:

- Name Server Set-up ?**: Includes a 'Server Key' field with the value '_u56sEubZjw', a 'Name Server URL' field with the value 'http://localhost/git/yioop2/', and a 'Use Filecache' checkbox with a '[Clear Cache]' link.
- Database Set-up ?**: Includes a 'Database System' dropdown menu set to 'Sqlite3' and a 'Database Name' field with the value 'public_default'.
- Account Registration ?**: Includes a 'Disable Registration' dropdown menu.
- Proxy Servers ?**: Includes a 'Tor Proxy (.onion urls only):' field with the value '127.0.0.1:9150' and a 'Crawl via Proxies' checkbox.
- Monetization ?**: Includes a 'Type' dropdown menu set to 'None'.
- Bot Configuration ?**: Includes a 'Bot Settings' dropdown menu set to 'Enable Bot Users'.

A 'Save' button is located at the bottom of the form. At the bottom of the page, there is a footer with links: '- Blog - Privacy - Terms - ThisSiteBot - Developed at SeekQuarry - (c) This Site - This Search Engine'.

The **Name Server Set-up** fieldset is used to tell Yioop which machine is going to act as a name server during a crawl and what secret string to use to make sure that communication



terms of the configuration settings of the indexer, you can choose anything for your secret string as long as you use the same string amongst all of the machines in your Yioop installation. The reason why you have to set the name server url is that each machine that is going to run a fetcher to download web pages needs to know who the queue servers are so they can request a batch of urls to download. There are a few different ways this can be set-up:

1. If the particular instance of Yioop is only being used to display search results from crawls that you have already done, then this fieldset can be filled in however you want.
2. If you are doing crawling on only one machine, you can put `http://localhost/path_to_yioop/` or `http://127.0.0.1/path_to_yioop/`, where you appropriately modify "path_to_yioop".
3. Otherwise, if you are doing a crawl on multiple machines, use the url of Yioop on the machine that will act as the name server.

In communicating between the fetcher and the server, Yioop uses curl. Curl can be particular about redirects in the case where posted data is involved. i.e., if a redirect happens, it does not send posted data to the redirected site. For this reason, Yioop insists on a trailing slash on your queue server url. Beneath the Queue Server Url field is a Filecache checkbox. Sometimes it is useful to be able while testing queries to remove what it currently cached, so that one can see new results that were just added, the **Clear Cache** link let's you do this. The **Clear Cache** link clears both the file cache as well as the local domain name to IP address cache.

The **Database Set-up** fieldset is used to specify what database management system should be used, how it should be connected to, and what user name and password should be used for the connection. At present [PDO](#) (PHP's generic DBMS interface), sqlite3, and Mysql databases are supported. Th demo site *yioop.com* uses the PDO interface to connect to a Postgres database. For a Yioop system, the database is used to store information about what users are allowed to use the admin panel and what activities and roles these users have. Unlike many database systems, if an sqlite3 database is being used then the connection is always a file on the current filesystem and there is no notion of login and password, so in this case only the name of the database is asked for. For sqlite, the database is stored in `WORK_DIRECTORY/data`. For single user settings with a limited number of news feeds, sqlite is probably the most convenient database system to use with Yioop. If you think you are going to make use of Yioop's social functionality and have many users, feeds, and crawl mixes, using a system like Mysql or Postgres might be more appropriate.

If you would like to use a different DBMS than Sqlite or Mysql, then the easiest way is to select PDO as the Database System and for the Hostname given use the DSN with the appropriate DBMS driver. For example, for Postgres one might have something like:

```
pgsql:host=localhost;port=5432;dbname=test;user=bruce;password=mypass
```

You can put the username and password either in the DSN or in the Username and Password fields. The database name field must be filled in with the name of the database you want to connect to. It is also include needs to be included in the dsn, as in the above. PDO and Yioop has been tested to work with Postgres and sqlite, for other DBMS's it might take some tinkering to get things to work.

When switching database information, Yioop checks first if a usable database with the user supplied data exists. If it does, then it uses it; otherwise, it tries to create a new database. Yioop comes with a small sqlite demo database in the data directory and this is used to populate the installation database in this case. This database has one account root with no password which has privileges on all activities. Since different databases associated with a Yioop installation might have different user accounts set-up after changing database information you might have to sign in again.

The **Account Registration** fieldset is used to control how user's can obtain accounts on a



up, Email Activation, after registering, users must click on a link which comes in a separate email to activate their accounts; and Admin Activation, after registering, an admin account must activate the user before the user is allowed to use their account. When Disable Registration is selected, the Suggest A Url form and link on the tool.php page is disabled as well, for all other registration type this link is enabled. If Email Activation is chosen, then the reset of this fieldset can be used to specify the email address that the email comes to the user. The Send Mail From Media Updater checkbox controls whether emails are sent immediately from the web app or if they are queued, and then sent out by the Media Updater process/processes. The checkbox Use PHP mail() function controls whether to use the mail function in PHP to send the mail, this only works if mail can be sent from the local machine. Alternatively, if this is not checked like in the image above, one can configure an outgoing SMTP server to send the email through.

The **Proxy Server** fieldset is used to control which proxies to use while crawling. By default Yioop does not use any proxies while crawling. A Tor Proxy can serve as a gateway to the Tor Network. Yioop can use this proxy to download .onion URLs on the [Tor network](#). The configuration given in the example above works with the Tor Proxy that comes with the [Tor Browser](#). Obviously, this proxy needs to be running though for Yioop to make use of it. A more robust approach than using the **Tor Browser** to relay, is to install the tor relay executable. From the command line you can install this using brew (MacOS), apt (Ubuntu Linux), or choco (Windows) using a command like:

```
brew install tor
```

Then use the setting **127.0.0.1:9050** in Yioop. Beneath the Tor Proxy input field is a checkbox labelled Crawl via Proxies. Checking this box, will reveal a textarea labelled Proxy Servers. You can enter the address:port, address:port:proxytype, or address:port:proxytype:username:password of the proxy servers you would like to crawl through. If proxy servers are used, Yioop will make any requests to download pages to a randomly chosen server on the list which will proxy the request to the site which has the page to download. To some degree this can make the download site think the request is coming from a different ip (and potentially location) than it actually is. In practice, servers can often use HTTP headers to guess that a proxy is being used.

The **Monetization** fieldset is used to specify whether advertising will be served with Yioop search results, and if it is, what kind. It also controls whether or not user's can charge credit fees for groups. The **Type** dropdown controls this, allowing the administrator to choose between **None** , **External Ad Server** , **Group Fees** , **Keyword Advertisements** , and **Group Fees and Keyword Ads** . If **Group Fees** or **Group Fees and Keyword Ads** are chosen then when a user creates a new group they can select for a registration type that a certain number of credits be charged. Yioop's built-in mechanism for keyword advertising is described in the [Keyword Advertising](#) section and is activated if either **Keyword Advertisements** or **Group Fees and Keyword Ads** is selected. **External Ad Server** refers to advertising scripts (such as Google Ad Words, Bidvertiser, Adspeed, etc) which are to be added on search result pages or on discussion thread pages. There are four possible placements of ads: None -- don't display advertising at all, Top -- display banner ads beneath the search bar but above search results, Side -- display skyscraper Ads in a column beside the search results, and Both -- display both banner and skyscraper ads. Choosing any option other than None reveals text areas where one can insert the Javascript one would get from the Ad network. The **Global Ad Script** text area is used for any Javascript or HTML the Ad provider wants you to include in the HTML head tag for the web page (many advertisers don't need this).

The **Bot Configuration** field-set is used to control whether user's of this Yioop instance can be chat bots. If enabled under Manage Accounts a Yioop user can declare themselves a chat bot. Such user's can be programmed to give automated responses to group posts using the [Chat Bot Interface](#) .

The **Security** activity looks like:



Session ?

Timezone:

Token Name:

Session Name:

Autologout:

Cookie Consent Expires:

Privacy ?

Differential Privacy:

Group Analytics:

Search Analytics:

Captcha Type ?

Recovery Type ?

[\[Edit Account Recovery Questions\]](#)

The **Session** fieldset is used to control how a user session in Yioop behaves. The **Timezone** field controls the timezone used for dating posts and other events once a user is logged in. The **Token Name** field controls the name of the token variable which appears in URLs that is used in conjunction with cookies to determine if a user is logged. It is there to prevent cross-site request forgery attacks on a Yioop website. The **Session Name** field controls the name of the cookie that will be stored in a user's browser to maintain a session once logged into Yioop. The **Autologout** dropdown specifies how long a session can be inactive before a user will be logged out. The **Cookie Consent Expires** field controls how long session cookies last before the message that a user needs to accept cookies to use the site fully appears. This message might be needed in some jurisdictions because of laws like [GDPR](#). The way this message is currently implemented in Yioop prevents the setting of any cookies in a user's browser until the consent message button is clicked.

The **Privacy Fieldset** controls a variety of options with respect to how analytics from individual users is collected by a Yioop instance. **Differential Privacy** controls whether group and thread view statistics which are displayed under the Manage Groups and Feeds and Wikis activities are fuzzified to try to prevent individual users from being identifiable by change in counts. **Group Analytics** controls whether information about group and thread views is collected and whether statistics about these views are visible to group owners. If this is disabled, it does not delete statistics that were previously collected, however, they will no longer be viewable and no future views will be recorded. **Search Analytics** controls whether information about search queries is collected and aggregated. If this is disabled, it does not delete statistics that were previously collected, however, they will no longer be viewable and no future collection will occur. Also, if this is disabled, but keyword advertisements are enabled, then impressions with respect to advertised keywords will still be collected.

The **Captcha Type** fieldset controls what kind of [captcha](#) will be used during account



series of dropdown answers to questions of the form: Which in the following list is the most/largest/etc? or Which is the following list is the least/smallest/etc?; Graphic Captcha, the user needs to enter a sequence of characters from a distorted image; and hash captcha, the user's browser (the user doesn't need to do anything) needs to extend a random string with additional characters to get a string whose hash begins with a certain lead set of characters. Of these, Hash Captcha is probably the least intrusive but requires Javascript and might run slowly on older browsers. A text captcha might be used to test domain expertise of the people who are registering for an account. Finally, the graphic captcha is probably the one people are most familiar with.

The **Recovery Type** fieldset controls the recovery option used when a user forgets their password. The choices are: **No Password Recovery Link**, in which case there is no automated option for recovering one's password; **Email Link Password Recovery**, in which case a link is sent to a user's email and clicking on that link allows the user to change their password; and **Email Link and Check Questions Recovery**, which is the same as Email Link Password Recovery except that to change the password the user must enter the answers to their recovery questions correctly.

The **Captcha and Recovery Questions** section of the Security activity provides links to edit the Text Captcha and Recovery Questions for the current locale (you can change the current locale in Settings). In both cases, there are a fixed list of tests you can localize. A single test consists of a more question, a less question, and a comma separate list of possibilities. For example,

```
Which lives or lasts the longest?  
Which lives or lasts the shortest?  
lightning,bacteria,ant,dog,horse,planet,star,galaxy
```

When challenging a user, Yioop picks a subset of tests. For each test, it randomly chooses between more or less question. It then picks a subset of the ordered list of choices, randomly permutes them, and presents them to the user in a dropdown.

Yioop's captcha-ing system tries to prevent attacks where a machine quickly tries several possible answers to a captcha. Yioop has a IP address based timeout system (implemented in models/visitor_model.php). Initially a timeout of one second between requests involving a captcha is in place. An error screen shows up if multiple requests from the same IP address for a captcha page are made within the time out period. Every mistaken entry of a captcha doubles this timeout period. The timeout period for an IP address is reset on a daily basis back to one second.

[Return to table of contents](#) .

Upgrading Yioop

If you have an older version of Yioop that you would like to upgrade, make sure to back up your data. Be advised that the upgrade from a version 2 to version 3 or higher of Yioop is a reasonably major upgrade, your indexes should work after the upgrade, but version 2 classifiers do not work in Yioop version 3. Yioop stores data in two places: a database, the details of which can be found by looking at the Server Settings activity; and in a WORK_DIRECTORY folder, the path to this folder can be found in the Configure activity, Search Engine Work Directory form. Once your back up is done, how you should upgrade depends on how you installed Yioop in the first place. We give three techniques below, the first of which will always work but might be slightly more complicated than the other two.

If you originally installed Yioop from the download page, download the latest version of Yioop. Extract the src subfolder of the zip file you get and replace the current src folder of the installation you have. If you had a file src/configs/LocalConfig.php, in your old installation make sure to copy this file to the upgrade. Make sure Yioop has write permissions on src/configs/Configs.php. Yioop should be then able to complete the upgrade process. If it doesn't look like your old work directory data is showing up, please check the Configure activity, Search Engine Work Directory form and make sure it has the same value as before the upgrade. In the case of upgrading a version 2 to a version 3 instance of Yioop, you should extract the whole Yioop folder you download and replace the



If you have been getting Yioop directly from the git repository, then to upgrade Yioop you can just issue the command:

```
git pull
```

from a command prompt after you've switched into the Yioop directory.

If you are using Yioop as part of a Composer project, then issue the command:

```
composer update
```

should upgrade Yioop to the most recent version compatible with your project.

[Return to table of contents](#) .

Summary of Files and Folders

The Yioop search engine consists of three main scripts:

src/executables/Fetcher.php

Used to download batches of urls provided the queue server.

src/executables/QueueServer.php

Maintains a queue of urls that are going to be scheduled to be seen. It also keeps track of what has been seen and robots.txt info. Its last responsibility is to create the `index_archive` that is used by the search front end.

index.php

Acts as the search engine web page. It is also used to handle message passing between the fetchers (multiple machines can act as fetchers) and the queue server. Finally, when Yioop is run as its own web server, this file launches and initializes the web server.

The file *index.php* is used when you browse to an installation of a Yioop website. The description of how to use a Yioop web site is given in the sections starting from the The Yioop User Interface section. The files *Fetcher.php* and *QueueServer.php* are only connected with crawling the web. If one already has a stored crawl of the web, then you no longer need to run or use these programs. For instance, you might obtain a crawl of the web on your home machine and upload the crawl to a an instance of Yioop on the ISP hosting your website. This website could serve search results without making use of either *Fetcher.php* or *QueueServer.php*. To perform a web crawl you need to use both of these programs as well as the Yioop web site. This is explained in detail in the section on [Performing and Managing Crawls](#) .

The Yioop folder itself consists of several files and sub-folders. The file *index.php* as mentioned above is the main entry point into the Yioop web application. There is also an *.htaccess* file that is used to route most requests to the Yioop folder to go through this *index.php* file. If you look at the top folder structure of Yioop it has three main subfolders: **src** , **work_directory** , and **tests** . If you are developing a [Composer project](#) with Yioop there might also be a **vendor** folder which contains a class autoloader for the PHP classes used in your project. The **src** folder contains the source code for Yioop. The **work_directory** is the default location where Yioop stores indexes as well as where you can put any site specific customizations. As the location of where Yioop stores stuff is customizable, we will write `WORK_DIRECTORY` in caps to refer to location of this folder in the current instance of Yioop. Finally, **tests** is a folder of unit tests for various Yioop classes. We now describe the major files and folders in each of these folders in a little more detail.

The **src** contains several useful files as well as additional sub-folders. *yioopbar.xml* is the xml file specifying how to access Yioop as an Open Search Plugin. *favicon.ico* is used to display the little icon in the url bar of a browser when someone browses to the Yioop site. A URL to the file *bot.php* is given by the Yioop robot as it crawls websites so that website owners can find out information about who is crawling their sites. Here is a rough guide to what the **src** folder's various sub-folders contain:



admin panel of Yioop. The file *Config.php* contains a number of parameters about how data is stored, how, and how often, the queue server and fetchers communicate, and which file types are supported by Yioop. *ConfigureTool.php* is a command-line tool which can perform some of the configurations needed to get a Yioop installation running. It is only necessary in some virtual private server settings -- the preferred way to configure Yioop is through the web interface. *Createdb.php* can be used to create a bare instance of the Yioop database with a root admin user having no password. This script is not strictly necessary as the database should be creatable via the Admin panel; however, it can be useful if the database isn't working for some reason. *Createdb.php* includes the file *PublicHelpPages.php* from *WORK_DIRECTORY/app/configs* if present or from *BASE_DIR/configs* if not. This file contains the initial rows for the Public and Help group wikis. When upgrading, it is useful to export the changes you have made to these wikis to *WORK_DIRECTORY/app/configs/PublicHelpPages.php*. This can be done by running the file *ExportPublicHelpDb.php* which is in the configs folder. Also, in the configs folder is the file *default_crawl.ini*. This file is copied to *WORK_DIRECTORY* after you set this folder in the admin/configure panel. There it is renamed as *crawl.ini* and serves as the initial set of sites to crawl until you decide to change these. The file *GroupWikiTool.php* can be used to determine where the resources for a wiki page are stored, as well as to do simple manipulations on the way the resources are archived when new versions of a wiki page are made. Running it with no command line arguments gives a description of how it works. The file *TokenTool.php* is a tool which can be used to help in term extraction during crawls and for making trie's which can be used for word suggestions for a locale. To help word extraction this tool can generate in a locale folder (see below) a word bloom filter. This filter can be used to segment strings into words for languages such as Chinese that don't use spaces to separate words in sentences. For trie and segmenter filter construction, this tool can use a file that lists words one on a line. *TokenTool.php* can also be used with [Yandex translate](#) to translate locale strings, Public and Help wiki pages from the English locale to other locales.

controllers

The controllers folder contains all the controller classes used by the web component of the Yioop search engine. Most requests coming into Yioop go through the top level index.php file. The query string (the component of the url after the ?) then says who is responsible for handling the request. In this query string there is a part which reads *c=...* This says which controller should be used. The controller uses the rest of the query string such as the *a=* variable for activity function to call and the *arg=* variable to determine which data must be retrieved from which models, and finally which view with what elements on it should be displayed back to the user. Within the controller folder is a sub-folder *components*, a component is a collection of activities which may be added to a controller so that it can handle a request.

css

This folder contains the stylesheets used to control how web page tags should look for the Yioop site when rendered in a browser.

data

This folder contains a default sqlite database for a new Yioop installation. Whenever the *WORK_DIRECTORY* is changed it is this database which is initially copied into the *WORK_DIRECTORY* to serve as the database of allowed users for the Yioop system.

examples

This folder contains files *QueryCacher.php*, *SearchApi.php*, *StockBot.php*, and *WeatherBot.php*. *QueryCacher.php* gives an example of how to write a script that executes a sequence of queries against a Yioop index. This could be useful for caching queries to improve query performance. *SearchApi.php* gives an example of how to use the Yioop search function api. Finally, *StockBot.php* and *WeatherBot.php* give examples of how to code a Yioop Chat Bot.



already mentioned, it contains: *ArcTool.php* , *ClassifierTool.php* , *ClassifierTrainer.php* , *CodeTool.php* , *Mirror.php* , *MediaUpdater.php* , and *QueryTool.php* . *ArcTool.php* can be used to examine the contents of WebArchiveBundle's and IndexArchiveBundle's from the command line. *ClassifierTool.php* is a command line tool for creating a classifier it can be used to perform some of the tasks that can also be done through the [Web Classifier Interface](#) . *ClassifierTrainer.php* is a daemon used in the finalization stage of building a classifier. *CodeTool.php* is for use by developers to maintain the Yioop code-base in various ways. *Mirror.php* can be used if you would like to create a mirror/copy of a Yioop installation. *MediaUpdater.php* can be used to do hourly updates of news feed search sources in Yioop. It also does video conversions of video files into web formats. Finally, *QueryTool.php* can be used to run queries from the command-line.

library

This folder is short for library. It contains all the common classes for things like indexing, storing data to files, parsing urls, etc. lib contains eight subfolders: *archive_bundle_iterators* , *classifiers* , *compressors* , *index_bundle_iterators* , *indexing_plugins* , *media_jobs* , *processors* , *summarizers* . The *archive_bundle_iterators* folder has iterators for iterating over the objects of various kinds of web archive file formats, such as arc, wiki-media, etc. These iterators are used to iterate over such archives during a recrawl. The classifier folder contains code for training classifiers used by Yioop. The *compressors* folder contains classes that might be used to compress objects in a web_archive. The *index_bundle_iterators* folder contains a variety of iterators useful for iterating over lists of documents which might be returned during a query to the search engine. The *media_jobs* folder contains subclasses of MediaJob. These are jobs which are run on a periodic basis by the MediaUpdater class to do things such as update news feeds in the index, send bulk emails about group activities, compute group and query analytics, and to convert video files. The *processors* folder contains processors to extract page summaries for a variety of different mimetypes. The *summarizers* folder contains summarizers which could be used when processing a page to be indexed to produce a summary of what should be indexed out of that page.

locale

This folder contains the default locale data which comes with the Yioop system. A locale encapsulates data associated with a language and region. A locale is specified by an [IETF language tag](#) . So, for instance, within the locale folder there is a folder en-US for the locale consisting of English in the United States. Within a given locale tag folder there is a file *configure.ini* which contains translations of string ids to string in the language of the locale. This approach is the same idea as used in [Gettext](#) .po files. Yioop's approach does not require a compilation step nor a restart of the webserver for translations to appear. On the other hand, it is slower than the Gettext approach. Besides the file *configure.ini*, there is a *statistics.txt* file which has info about what percentage of the id's have been translated. In addition to *configure.ini* and *statistics.txt*, the locale folder for a language contains a *resources* subfolder. The resources folder contains the files: *locale.js* , which contains locale specify Javascript code such as the variable *alpha* which is used to list out the letters in the alphabet for the language in question for spell check purposes, and *roman_array* for mapping between roman alphabet and the character system of the locale in question; *suggest-trie.txt.gz* , a Trie data structure used for search bar word suggestions; and *Tokenizer.php* , which can specify the number of characters for this language to constitute a char gram, might contain *segmenter* to split strings into words for this language, a *stemmer* class used to stem terms for this language, a *stopword remover* for the centroid summarizer, or a part of *speech tagger* for the locale.

models

This folder contains the subclasses of Model used by Yioop Models are used to



datasources folder. A **datasource** is an abstraction layer for the particular filesystem and database system that is being used by a Yioop installation. At present, **datasources** have been defined for PDO (PHP's generic DBMS interface), **sqlite3**, and **mysql** databases.

resources

Used to store binary resources such as graphics, video, or audio. For now, just stores the Yioop logo.

scripts

This folder contains the Javascript files used by Yioop.

views

This folder contains View subclasses as well as folders for elements, helpers, and layouts. A View is responsible for taking data given to it by a controller and formatting it in a suitable way. Most views output a web page; however, some of the views responsible for communication between the fetchers and the queue server output serialized objects. The elements folder contains Element classes which are typically used to output portions of web pages. For example, the **html** that allows one to choose an Activity in the Admin portion of the website is rendered by an **ActivityElement**. The helpers folder contains Helper subclasses. A Helper is used to automate the task of outputting certain kinds of web tags. For instance, the **OptionsHelper** when given an array can be used to output select tags and option tags using data from the array. The layout folder contains Layout subclasses. A Layout encapsulates the header and footer information for the kind of a document a View lives on. For example, web pages on the Yioop site all use the **WebLayout** class as their Layout. The **WebLayout** class has a render method for outputting the doctype, open html tag, head of the document including links for style sheets, etc. This method then calls the render methods of the current View, and finally outputs scripts and the necessary closing document tags.

The second top level subfolder of a Yioop instance is the **tests** folder. This folder contains **UnitTest**'s and **JavascriptUnitTests** for various library and script components. Yioop comes with its own minimal **UnitTest** and **JavascriptUnitTest** classes which defined in the **src/library/Unit_Test.php** and **src/library/JavascriptUnitTest.php**. It also contains a few files used for experiments. For example, **StringCatExperiment.php** was used to test which was the faster way to do string concatenation in PHP. **ManyUserExperiment.php** can be used to create a test Yioop installation with many users, roles, and groups. Some unit testing of the wiki Help system makes use of [PhantomJS](#). If **PhantomJS** is not configured, these tests will be skipped. To configure **PhantomJS** you simply add a define for your path to **PhantomJS** to your **src/configs/LocalConfig.php** file. For example, one might have add the define: `define("PHANTOM_JS", "/usr/local/bin/phantomjs");`

The last top level folder that Yioop makes use of is the **WORK DIRECTORY**. The location of this directory is set during the configuration of a Yioop installation, but defaults to your yioop folder's **work_directory** sub-folder. Yioop stores crawls, and other data local to a particular Yioop installation in files and folders in this directory. In the event that you upgrade your Yioop installation you should only need to replace the Yioop **src** folder and in the configuration process of Yioop tell it where your **WORK DIRECTORY** is. Of course, it is always recommended to back up one's data before performing an upgrade. Within the **WORK DIRECTORY**, Yioop stores four main files: **Profile.php**, **crawl.ini**, **bot.txt**, and **robot_table.txt**. Here is a rough guide to what the **WORK DIRECTORY**'s sub-folder contain:

app

This folder is used to contain your overrides to the views, controllers, models, resources, locale etc. For example, if you wanted to change how the search results were rendered, you could add a **views/SearchView.php** file to the **app** folder and Yioop would use it rather than the one in the Yioop base directory's **views** folder. Using the **app dir** makes it easier to have customizations that won't get messed up when you



DoubleIndexDataUNIX_TIMESTAMP, IndexDataUNIX_TIMESTAMP, and QueueBundleUNIX_TIMESTAMP. ArchiveUNIX_TIMESTAMP folders hold complete caches of web pages that have been crawled. These folders will appear on machines which are running fetcher.php. IndexDataUNIX_TIMESTAMP folders hold a word document index as well as summaries of pages crawled. A folder of this type is needed by the web app portion of Yioop to serve search results. These folders can be moved from machine to whichever machine you want to server results from. A DoubleIndexDataUNIX_TIMESTAMP has two folders for a word document index as well as summaries of pages crawled - one active, one previous -- used to store repeating crawls. QueueBundleUNIX_TIMESTAMP folders are used to maintain the priority queue during the crawling process. The QueueServer.php program is responsible for creating both IndexDataUNIX_TIMESTAMP and QueueBundleUNIX_TIMESTAMP folders.

data

If an sqlite or sqlite3 (rather than say MySQL) database is being used then a default.db file is stored in the data folder. In Yioop, the database is used to manage users, roles, locales, and crawls. Data for crawls themselves are NOT stored in the database. Suggest a url data is stored in data in the file suggest_url.txt, certain cron information about machines is saved in cron_time.txt, and plugin configuration information can also be stored in this folder.

locale

This is generally a copy of the locale folder mentioned earlier. In fact, it is the version that Yioop will try to use first. It contains any customizations that have been done to locale for this instance of Yioop. If you using a version of Yioop after Yioop 2.0, this folder have been moved to app/locale.

log

When the fetcher and queue server are run as daemon processes log messages are written to log files in this folder. Log rotation is also done. These log files can be opened in a text editor or console app.

query

This folder is used to stored caches of already performed queries when file caching is being used.

schedules

This folder has four kinds of subfolders: media_convert, IndexDataUNIX_TIMESTAMP, RobotDataUNIX_TIMESTAMP, and ScheduleDataUNIX_TIMESTAMP. The easiest to explain is the media_convert folder. It is used by MediaUpdater.php to stored job information about video files that need to be converted. For the other folder, when a fetcher communicates with the web app to say what it has just crawled, the web app writes data into these folders to be processed later by the queue server. The UNIX_TIMESTAMP is used to keep track of which crawl the data is destined for. IndexData folders contain mini-inverted indexes (word document records) which are to be added to the global inverted index (called the dictionary) for that crawl. RobotData folders contain information that came from robots.txt files. Finally, ScheduleData folders have data about found urls that could eventually be scheduled to crawl. Within each of these three kinds of folders there are typical many sub-folders, one for each day data arrived, and within these subfolders there are files containing the respective kinds of data.

temp

This is used for storing temporary files that Yioop creates during the crawl process. For example, temporary files used while making thumbnails. Each fetcher has its own temp folder, so you might also see folders 0-temp, 1-temp, etc.

[Return to table of contents](#) .

Search and User Interface

to perform crawls with Yioop in more detail in the [Crawling and Customizing Results](#) chapter. If you do not have a crawl available, you can test some of these features on the [Yioop Demo Site](#).

Search Basics

The main search form for Yioop looks like:

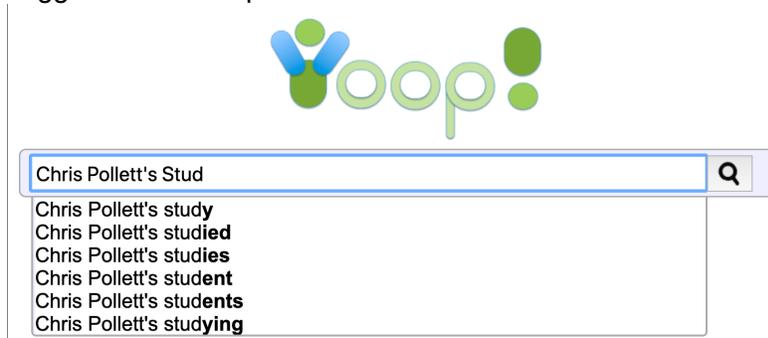


The screenshot shows the Yioop search interface. At the top is the Yioop logo, which consists of three stylized human figures in blue and green. Below the logo is a search input field with the placeholder text "Type what to find" and a search button with a magnifying glass icon. Below the search field are three sections of suggestions:

- Trends...**
 - [stock market](#) - [stay-at-home](#) - [Donald Trump](#) -
 - [Supreme Court](#) - [Covid-19](#) - [Attorney General](#) -
 - [national security](#) - [police officer](#) - [COVID-19](#) -
- News...**
 - [Artist replaces slave trader statue with one of a Black...](#)
CNN
 - [Trump retweets ex-game show host comments that CDC is...](#)
Yahoo News
 - [Flooded southern China braces for more storms...](#)
Yahoo News
 - [Apple wins EU court case over \\$15 billion in claimed taxes...](#)
Associated Press Top Headlines
 - [Assam: Fifty dead and two million hit by India monsoon...](#)
Yahoo News
- Sports...**
 - [Bolt contemplates coach trip to Olympics...](#)
Latest News Now Athletics
 - [Who Will Be MLB's Worst Team in 2020?](#)
Sports Illustrated
 - [Napoli set to sell Allan...](#)
Soccer News
 - [These SQAIRZ shoes are big on comfort and stability to keep...](#)
Golf.com
 - [2020 Memorial Tournament DFS Plays: Cash Plays, Tournament...](#)
Sports Illustrated

At the bottom of the page, there is a footer: [Blog](#) - [Privacy](#) - [Terms](#) - [YioopBot](#) - Developed at [SeekQuarry](#) - (c) 2020 Yioop - PHP Search Engine

The HTML for this form is in `views/elements/SearchElement.php` and the icon is stored in `resources/yioop.png`. You may want to modify these to incorporate Yioop search into your site. For more general ways to modify the look of this pages, consult the [Building a site using Yioop](#) documentation. The Yioop logo on any screen in the Yioop interface is clickable and returns the user to the main search screen. One performs a search by typing a query into the search form field and clicking the Search button. As one is typing, Yioop suggests possible queries, you can click, or use the up down arrows to select one of these suggestion to also perform a search



The screenshot shows the Yioop search interface with a search input field containing the text "Chris Pollett's Stud". Below the search field is a dropdown menu with the following suggestions:

- Chris Pollett's study
- Chris Pollett's studied
- Chris Pollett's studies
- Chris Pollett's student
- Chris Pollett's students
- Chris Pollett's studying

For some non-roman alphabet scripts such as Telugu you can enter words using how they sound using roman letters and get suggestions in the script in question:

- తెలంగాణ
- తెలంగాణ చరిత్ర
- తెలంగాణ పరిస్థితులు
- తెలంగాణ సాహిత్యం
- తెలంగాణ యాత్ర
- తెలంగాణ పుస్తకాలు

The [More Statistics] link only shows if under the Admin control panel you clicked on more statistics for the crawl. This link goes to a page showing many global statistics about the web crawl. Beneath this link are the Blog and Privacy links (as well as a link back to the SeekQuarry site). These two links are to static pages which can be customized through the Manage Locale activity. Typical search results might look like:



The world will see (and just saw) an Internet zombie apocalypse - Phil Stanhope (Dyn) - O'Reilly Sec (15 Results)



www.oreilly.com/library/view/ore...119.html Words: reilly conference 2016 amsterdam netherlands Selection from O'Reilly Security Conference 2016 - Amsterdam, Netherlands [Video] . The world will see (and just saw) an Internet zombie

Zombie apocalypse

In some stories, victims of zombies may become zombies themselves if they are bitten by zombies or if a zombie-creating virus travels by air, sexually, or by water; in others, everyone who dies, whatever the cause, becomes one of the undead. More at Wikipedia...

Using your Maker Skills to survive a Zombie Apocalypse with Raspberry Pi and Arduino - Using your Ma



www.oreilly.com/library/view/usi...332.html Words: maker raspberry arduino pi skills Selection from Using your Maker Skills to survive a Zombie Apocalypse with Raspberry Pi and Arduino [Video] . Using your Maker Skills to

Steam Community :: They Are Billions

steamcommunity.com/app/644930 Words: billions infected game human build They Are Billions - They Are Billions is a strategy game in a distant future about building and managing human colonies after a zombie apocalypse

Preparedness 101: Zombie Apocalypse - Wikipedia

en.m.wikipedia.org/wiki/2011_CDC...ocalypse Words: cdc zombie apocalypse post edit Preparedness 101: Zombie Apocalypse " is a blog post by the United States Centers for Disease Control and Prevention (CDC) that uses a zombie

Free and Premium WordPress Themes - Croyout Creations

www.croyoutcreations.eu Words: wordpress themes free creations theme Our WordPress themes are some of the most customizable themes on WordPress.org. Designed with great care for authors, blogs and businesses alike.

Resident Evil (film series) - Wikipedia

en.m.wikipedia.org/wiki/Resident...ndtrack Words: evil resident film anderson alice Resident Evil is an action horror science fiction film series loosely based on the Capcom survival horror video game franchise of the same name .

Train to Busan - Wikipedia

en.m.wikipedia.org/wiki/Train_to_Busan Words: train film nominated busan best Train to Busan (Korean :&p; 부산행 ; Hanja :&p; 釜山行 ; RR :&p; Busanhaeng) is a 2016 South Korean action-horror film directed by Yeon Sang-ho and

Apocalyptic and post-apocalyptic fiction - Wikipedia

en.m.wikipedia.org/wiki/Post-apocalyptic Words: article issues multiple learn template This article has multiple issues. Please help improve it or discuss these issues on the talk page . (Learn how and when to remove these template

Apocalyptic and post-apocalyptic fiction - Wikipedia

en.m.wikipedia.org/wiki/Apocalyp...fiction Words: article issues multiple learn template This article has multiple issues. Please help improve it or discuss these issues on the talk page . (Learn how and when to remove these template

Shaun of the Dead - Wikipedia

en.wikipedia.org/wiki/Shawn_of_the_Dead Words: film shaun wright zombie 2004 zombie comedy film directed by Edgar Wright. Shaun of the Dead is a 2004 horror comedy film directed by Edgar Wright . The film was written by

The Cured - Wikipedia

en.m.wikipedia.org/wiki/The_Cured Words: cured senan infected conor contents The Cured is a 2017 Irish horror drama film written and directed by David Freyne in his feature debut film. It stars Ellen Page , Sam Keeley , and

Category:2015 British television series endings - Wikipedia

en.wikipedia.org/wiki/Category:2...endings Words: series tv 2015 united kingdom Television portal United Kingdom portal Television series which originated in the United Kingdom and ended in the year 2015 . Shows that

"The Blacklist," other TV adapt to coronavirus. What's next? - Los Angeles Times

www.latimes.com/entertainment-ar...ght-zone Words: tv television blacklist times pandemic Parks and Recreation, "All Rise," "The Blacklist" and other series have recalibrated to life in a pandemic. But what will scripted TV look like

Review: Universal hones Horror Nights into a well-oiled killing machine - Los Angeles Times

www.latimes.com/travel/themepark...ony.html Words: horror nights universal killing machine After years of simulated neck-slicing and chest-stabbing mayhem, Universal Studios Hollywood has fine-tuned its Halloween Horror Nights into a

overview for Numi18

www.reddit.com/user/Numi18/?sort=hot Words: reddit numi18 community points save The u/Numi18 community on Reddit. Reddit gives you the best of the internet in one place. overview for Numi18 jump to content my subreddits edit

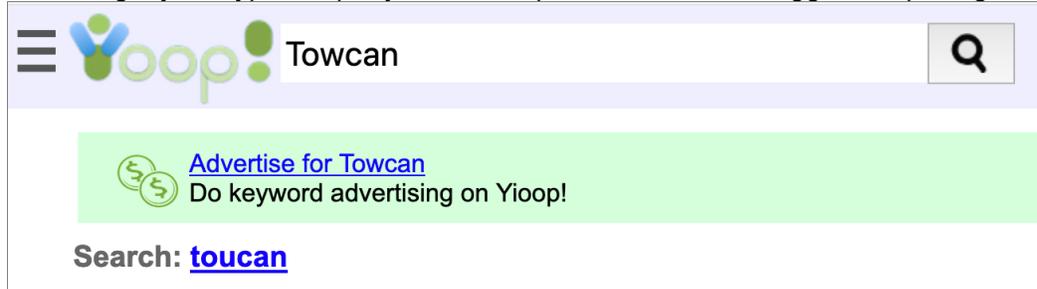
Next

The terms next to "Words:" are a word cloud of important terms in the document. Hovering over the Score of a search result reveals its component scores. These might include: Rank, Relevance, Proximity, as well as any Use to Rank Classifier scores.

Rel:50.74
Prox:1.00

skin trending sea

If one slightly mistypes a query term, Yioop can sometimes suggest a spelling correction:



The screenshot shows the Yioop search interface. At the top left is a menu icon. The search bar contains the text 'Towcan' and a search icon. Below the search bar is a green banner with a dollar sign icon and the text 'Advertise for Towcan Do keyword advertising on Yioop!'. Below the banner, the search results are displayed as 'Search: [toucan](#)'.

Each result back from the query consists of several parts: First comes a title, which is a link to the page that matches the query term. This is followed by a brief summary of that page with the query words in bold. Then the document rank, relevancy, proximity, and overall scores are listed. Each of these numbers is a grouped statistic -- several "micro index entries" are grouped together/summed to create each. So even though a given "micro index entry" might have a document rank between 1 and 10 there sum could be a larger value. Further, the overall score is a generalized inner product of the scores of the "micro index entries", so the separated scores will not typically sum to the overall score. After these scores there are three links: Cached, Similar, and Inlinks. Clicking on Cached will display Yioop's downloaded copy of the page in question. We will describe this in more detail in a moment. Clicking on Similar causes Yioop to locate the five words with the highest relevancy scores for that document and then to perform a search on those words. Clicking on Inlinks will take you to a page consisting of all the links that Yioop found to the document in question. Finally, clicking on an IP address link returns all documents that were crawled from that IP address.

For some languages such as English, Yioop supports a question answer subsystem. On a query, such as "Who is Justin Trudeau?", Yioop next to each result might display a possible answer to the query. This looks like

[Highlights of Trudeau's budget interview with Global BC | Watch News Videos Online](#)
[globalnews.ca/video/2596462/high...lobal-bc](#) Possible Answer: the prime minister Words: global bc morning news trudeau
 Highlights of Trudeau s budget interview with Global BC. ... BC. Highlights of Trudeau s budget interview with Global BC. Wed ... , Mar 23: Prime Minister Justin Trudeau comments on how the
[Cached](#).

Below is an example how the Cache of a page might look in Yioop



The screenshot shows a cached version of a page. At the top, it says 'This cached version of <http://www.ucanbuyart.com/> was obtained by the Yioop crawler on December 17 2012 10:30:02.' Below this is a 'Toggle History' link. Another 'Toggle Extracted Headers and Summaries' link is visible. The main content of the page is the header for 'An Online Fine Art Gallery UCanBuyArt.com'. The header includes the text 'An Online Fine Art Gallery' and the logo 'UCanBuyArt.com'. Below the header, there are links for 'Art Resources', 'Join', 'About', 'Links', and a shopping cart icon.

As the above illustrates, on a cache link click, Yioop will display a cached version of the page. The cached version has a link to the original version and download time at the top. Next there is a link to display all caches of this page that Yioop has in any index. This is followed by a link for extracted summaries, then in the body of the cached document the query terms are highlighted. Links within the body of a cache document first target a cached version of the page that is linked to which is as near into the future of the current cached page as possible. If Yioop doesn't have a cache for a link target then it goes to location pointed to by that target. Clicking on the history toggle, produces the following



Year: 2013 | Month: January

• January 03 11:13

Toggle Extracted Headers and Summaries

An Online Fine Art Gallery



Art Resources Join About Links

This lets you select different caches of the page in question. Clicking the "Toggle extracted summary" link will show the title, summary, and links that were extracted from the full page and indexed. No other terms on the page are used to locate the page via a search query. This can be viewed as an "SEO" view of the page.

```
This cached version of http://www.ucanbuyart.com/ was obtained by the Yisooq crawler on January 03 2013 11:13:37.
Toggle History

Toggle Extracted Headers and Summaries

* About to connect() to 174.120.10.130 port 80 (#0)
* Trying 174.120.10.130...
* Connected to 174.120.10.130 (174.120.10.130) port 80 (#0)
> GET / HTTP/1.1
Range: bytes=0-50000
User-Agent: Mozilla/5.0 (compatible; YisooqBot; http://localhost/git/yisooq/bot.php)
Accept: */*
Accept-encoding: deflate, gzip
Host: www.ucanbuyart.com

HTTP/1.1 206 Partial Content
Date: Thu, 03 Jan 2013 19:13:37 GMT
Server: Apache
Cache-Control: max-age=7200, must-revalidate
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Range: bytes 0-6012/6013
Content-Length: 6013
Content-Type: text/html
X-Pad: avoid browser bug
Extracted Title
An Online Fine Art Gallery U Can Buy Art - Buy Fine Art Online

Extracted Description
.. UCanBuyArt.com is an online art gallery and dealer designed for sales of high quality and original fine renowned artists. categories include oil paintings, water colors, ceramics, artist bears, fantasy, Inuit, native, home decor. . . An Online Fine Art Gallery U Can Buy Art is a fine art dealer & gallery for famous artists original oil paintings, fine prints, watercolors, sculptures, photography, drawings, and Inuit. UCanBuyArt.com is designed with the philosophy that the fine arts and artists should be supported on a global medium and made accessible to all. With this in mind we have online community for artists to exhibit and sell their work directly to the public. Prices are in US dollars and purchases may be made directly from this secure online gallery using VISA or Master Card. Visit us often for the latest updates. Items range in price from $8 to $249,000. See our Fine Art Services for fine appraisals, professional restoration, upcoming shows, and much more! For a feeling of Christmas year round see our Chris .. Buy that perfect piece of fine art. Search our fine art gallery by one of the convenient options below. -Art Category -Artist Name -Gallery Advanced Search -Art Classifieds What's New! Site Index Contact Us link to Us Receive our newsletter! Online art gallery .. Buy that perfect piece of fine art. Search our fine art gallery by one of the convenient options below. -Art Category -Artist Name -Gallery Advanced Search -Art Classifieds What's New! Site Index Contact Us link to Us Receive our newsletter! Online art gallery .. Buy that perfect piece of fine art. Search our fine art gallery by one of the convenient options below.

Extracted Links
Array
(
    [http://www.ucanbuyart.com/default.html] =>
    [http://www.ucanbuyart.com/artresources.html] => Art Resources
    [http://www.ucanbuyart.com/join.html] => Join
    [http://www.ucanbuyart.com/about.html] => About
    [http://www.ucanbuyart.com/links.html] => Links
    [http://www.ucanbuyart.com/advancedsearch.html] => Advanced Search
    [http://www.ucanbuyart.com/classified.html] => Art Classifieds
    [http://www.ucanbuyart.com/new/] => What's New!
    [http://www.ucanbuyart.com/index.html] => Site Index
    [http://www.ucanbuyart.com/contact.html] => Contact Us
    [http://www.ucanbuyart.com/links.html] => Link to Us
    [http://www.ucanbuyart.com/newsletter.html] => Receive our newsletter!
    [http://www.ucanbuyart.com/mail_paintings.html] => oil paintings
    [http://www.ucanbuyart.com/art_prints.html] => fine prints
    [http://www.ucanbuyart.com/water_colors.html] => watercolors
    [http://www.ucanbuyart.com/sculptures.html] => sculptures
    [http://www.ucanbuyart.com/art_photography.html] => photography
    [http://www.ucanbuyart.com/art_drawings.html] => drawings
    [http://www.ucanbuyart.com/fine_art_services.html] => Fine Art Services
    [http://www.ucanbuyart.com/art_search.html] => fine appraisals
    [http://www.ucanbuyart.com/restoration.html] => professional restoration
    [http://www.ucanbuyart.com/events.html] => upcoming shows
    [http://www.ucanbuyart.com/christmas-boutique.html] => Christmas
    [] => sales@UCanBuyArt.com
    [http://www.ucanbuyart.com/Custom_Art.html] => View artist's who offer custom
    [http://www.ucanbuyart.com/fine_art.html] => Complete art gallery tour (alphabetical)
    [http://www.ucanbuyart.com/fine_art_alpha.html] => Complete art gallery tour (by price)
    [http://www.ucanbuyart.com/fine_art/logo1.jpg] => online fine art gallery u can buy art
    [http://www.ucanbuyart.com/images/visa_62x38.gif] => Visa
    [http://www.ucanbuyart.com/images/mc_62x38.gif] => Mastercard
)
```

An Online Fine Art Gallery



Art Resources Join About Links

It should be noted that cached copies of web pages are stored on the fetcher which originally downloaded the page. The IndexArchive associated with a crawl is stored on the queue server and can be moved around to any location by simply moving the folder. However, if an archive is moved off the network on which fetcher lives, then the look up of a cached page might fail. In addition, to a straightforward web search, one can also do image, video, news, and trends searches by clicking on the hamburger menu button and then clicking Images,



[What you need to know: Coronavirus and protective eyewear Video - ABC News](#) (61977 Results)



[abcnews.go.com/Health/video/coro...71258042](#) Words: **news video abc rights playing**
Dr. Sunir Garg, a retina surgeon and professor of ophthalmology, discusses how to protect your eyes from COVID-19. What you need to know:
Similar. [Inlinks.](#) [IP:143.204.131.93.](#) Score:9.8943

Coronavirus

Coronaviruses are species of virus belonging to the subfamily Coronavirinae in the family Coronaviridae, in the order Nidovirales.
[More at Wikipedia...](#)

[Coronavirus cases rise across the country Video - ABC News](#)



[abcnews.go.com/WNT/video/coronav...71248096](#) Words: **CASES news video coronavirus florida**
Florida hit a record of 2,000 cases in 24 hours for the second straight day, with other states hitting new highs.
Coronavirus cases rise across
Similar. [Inlinks.](#) [IP:143.204.131.93.](#) Score:9.8001

[Study shows possible new lifesaving coronavirus drug in UK Video - ABC News](#)



[abcnews.go.com/WNT/video/study-s...71286685](#) Words: **coronavirus study news steroid video**
A widely available and cheap steroid that reduces inflammation cut coronavirus deaths by one-third in an Oxford University study. Study shows
Similar. [Inlinks.](#) [IP:143.204.131.83.](#) Score:9.7208

[Coronavirus 'Carmageddon' coming? Video - ABC News](#)



[abcnews.go.com/US/video/coronavi...71267548](#) Words: **news coming abc coronavirus traffic**
ABC News' Gio Benitez reports on warnings that reduced use of mass transit could lead to increased traffic congestion as more Americans return to
Similar. [Inlinks.](#) [IP:143.204.131.83.](#) Score:9.6926

[Officials adjust plans amid new coronavirus hospitalizations Video - ABC News](#)



[abcnews.go.com/WNT/video/officia...71265006](#) Words: **news fda cases hydroxychloroquine video**
The FDA has issued a new warning regarding hydroxychloroquine as Miami and other cities slam the brakes on reopening after an alarming surge in
Similar. [Inlinks.](#) [IP:143.204.131.66.](#) Score:9.6894

[Rep. Val Demings says Florida's 'had some issues' with coronavirus response Video - ABC News](#)



[abcnews.go.com/theview/video/rep...71259570](#) Words: **florida news covid-19 video trump**
The Florida congresswoman addresses the rising number of COVID-19 cases in her state and if Jacksonville can handle Pres. Trump's upcoming rally.
Similar. [Inlinks.](#) [IP:143.204.131.93.](#) Score:9.5918

[COVID-19 drug breakthrough? Video - ABC News](#)



[abcnews.go.com/Health/video/covi...71289724](#) Words: **news drug covid-19 breakthrough coronavirus**
Dr. Simone Wildes discusses a new study from Oxford scientists that shows an inexpensive steroid already on the market may help treat the
Similar. [Inlinks.](#) [IP:143.204.131.83.](#) Score:9.2445

[Heroes return to work amid the pandemic Video - ABC News](#)



[abcnews.go.com/WNT/video/heroes...71265048](#) Words: **news coronavirus playing video mta**
An MTA worker in New York City and a nurse in Boston have recovered from the coronavirus and returned to the front lines. Heroes return to work
Similar. [Inlinks.](#) [IP:143.204.131.93.](#) Score:9.2065

[Vice President Pence says second wave concerns are 'overblown' Video - ABC News](#)



[abcnews.go.com/WNN/video/vice-pr...71293997](#) Words: **news 20 coronavirus rise cases**
This comes as 20 states are seeing a rise in coronavirus cases. ABC News' Alex Presha reports. Vice President Pence says second wave concerns
Similar. [Inlinks.](#) [IP:143.204.131.83.](#) Score:9.1334

[ABC News Daily: June 17, 2020 Video - ABC News](#)



[abcnews.go.com/US/video/abc-news...71045364](#) Words: **news stories abc catch making**
Catch up on the developing stories making headlines. ABC News Daily: June 17, 2020 Video - ABC News . Sections
Sections Top Stories Video Live
Similar. [Inlinks.](#) [IP:143.204.131.93.](#) Score:9.1156

Next



[This Tennessee doctor caught coronavirus at a meeting about coronavirus. He nearly died.](#) (8876 Results)
Yahoo News - 14/07/2020
[news.yahoo.com/tennessee-doctor-caught-coronavirus-meeting-113754285.html](#)
Dr. Daniel Lewis, the chief medical officer of a hospital in East Tennessee, nearly died after catching coronavirus. His recovery took

Coronavirus

Coronaviruses are species of virus belonging to the subfamily Coronavirinae in the family Coronaviridae, in the order Nidovirales.
[More at Wikipedia...](#)

[Bottleneck for US Coronavirus Response: The Fax Machine.](#) Slashdot - 13/07/2020
[rss.slashdot.org/~r/Slashdot/slashdot/~3/_PDT2qh5ivc/bottleneck-for-us-coronavirus...onse-the-fax-machine](#)
Public health officials in Houston are struggling to keep up with one of the nation's largest coronavirus outbreaks. They are desperate to trace

[Baby Was Infected With Coronavirus In Womb, Study Reports.](#) Slashdot - 14/07/2020
[rss.slashdot.org/~r/Slashdot/slashdot/~3/0w7FtkOHNE/baby-was-infected-with-coronavirus-in-womb-study-reports](#)
An anonymous reader quotes a report from The New York Times: Researchers on Tuesday [reported in the journal Nature Communications] strong evidence



[Bangladesh hospital owner arrested over fake coronavirus test results.](#) Yahoo News - 14 hours ago
[news.yahoo.com/bangladesh-hospital-owner-arrested-over-034011501.html](#)
A Bangladesh hospital owner accused of issuing thousands of fake negative coronavirus test results to patients at his two clinics was arrested on



[Masks prevented major coronavirus outbreak at hair salon: study.](#) Yahoo News - 14/07/2020
[news.yahoo.com/masks-prevented-major-coronavirus-outbreak-hair-salon-study-202922128.html](#)
Two US hair stylists who wore masks while infected with the coronavirus did not pass on COVID-19 to nearly 140 clients they saw over the course of



[Hospitals told to send coronavirus data to Washington, not CDC.](#) Yahoo News - 15/07/2020
[news.yahoo.com/hospitals-told-send-coronavirus-data-064649924.html](#)
"The new faster and complete data system is what our nation needs to defeat the coronavirus," Department of Health and Human Services

[2021 Rose Parade canceled due to coronavirus pandemic.](#) Associated Press Top Headlines - 20 hours ago
[apnews.com/e4ee408148f29e255a779694162b06ad](#)
PASADENA, Calif. (AP) — The 2021 Rose Parade has been canceled because of the impact of the coronavirus pandemic on planning for the New Year's Day



[Disneyland Hong Kong to Close Again Amid Coronavirus Case Rise.](#) The Hollywood Reporter - 13/07/2020
[www.hollywoodreporter.com/news/disneyland-hong-kong-close-again-coronavirus-case-rise-1302988](#)
The news comes days after Disney partially reopened a large portion of Disney World in Florida despite novel coronavirus cases skyrocketing in the

[Can a pregnant woman spread the coronavirus to her fetus?](#) Associated Press Top Headlines - 9 hours ago
[apnews.com/3f8fb2d0c7a46a987f40160a42b39780](#)
Can a pregnant woman spread the coronavirus to her fetus? It's possible, but it seems to be relatively rare and scientists think they know why that



['This Is Not a Normal Recession': Banks Ready for Wave of Coronavirus Defaults.](#) Wall Street Journal: Markets - 14/07/2020
[www.wsj.com/articles/this-is-not-a-normal-recession-banks-ready-for-wave-of-coronavirus-defaults-11594746008](#)
The largest U.S. banks signaled that the worst of the coronavirus recession is yet to come, opting to stow away tens of billions of dollars to



[U.S. House Republican Morgan Griffith tests positive for coronavirus.](#) Yahoo News - 14/07/2020
[news.yahoo.com/u-house-republican-morgan-griffith-192915609.html](#)
Republican U.S. Representative Morgan Griffith has tested positive for the coronavirus, his office said on Tuesday, making him at least the 10th



[German study finds no evidence coronavirus spreads in schools.](#) Yahoo News - 13/07/2020
[news.yahoo.com/german-study-finds-no-evidence-164704005.html](#)
Schools do not play a major role in spreading the coronavirus, according to the results of a German study released on Monday. The study, the largest



[McConnell: No recovery bill without lawsuit protections for 'everyone related to the coronavirus'.](#) CNN - 13/07/2020
[rss.cnn.com/~r/rss/cnn_topstories/~3/ybuOKNNn6Oc/index.html](#)
A new plan under development by the White House and Senate Republicans to deal with the fallout from the coronavirus pandemic is expected to include



[Brazilian native leader fights off coronavirus to take on the President.](#) CNN - 13/07/2020
[rss.cnn.com/~r/rss/cnn_topstories/~3/nBM7JSRW_w/index.html](#)
Crisanto Rudzō Tseremeywá was in hospital on a ventilator when his mother died from the coronavirus. The leader of some of Brazil's indigenous people



[Coronavirus may force Trump nominating convention outdoors.](#) Yahoo News - 14/07/2020
[news.yahoo.com/coronavirus-may-force-trump-nominating-184410827.html](#)
Republicans are planning to move their convention formally nominating President Donald Trump for a second White House term from an indoor arena to an



[Huge protests rock several countries as coronavirus ignites rage against governments.](#) CNN - 22 hours ago
[rss.cnn.com/~r/rss/cnn_topstories/~3/MVdlehYdQM/index.html](#)
Major protests are breaking out against governments in several countries after coronavirus sparked fury over



precautions, the World Health



[Japan says coronavirus adds to security threat by China](#). Yahoo News - 13/07/2020
[news.yahoo.com/japan-says-coronavirus-adds-security-040833822.html](#)
 China is pushing harder to make territorial claims in the regional seas and even using the coronavirus pandemic to expand its influence and take



[Florida GOP lawmaker's claim about coronavirus stuns Brianna Keilar](#). CNN - 22 hours ago
[rss.cnn.com/~r/cnn_topstories/~3/KkNBaWF2UOA/anthony-sabatini-florida-is-doi...eilar-sot-nr-vpx.cnn](#)
 Florida Republican state lawmaker Rep. Anthony Sabatini, who has filed numerous lawsuits to prevent mask mandates in his state, tells CNN's Brianna



[Coronavirus: Breaking down the outbreaks in Northern and Southern California](#). San Jose Mercury News - 1 hour ago
[www.mercurynews.com/2020/07/16/coronavirus-breaking-down-the-outbreaks-northern-...southern-california](#)
 Cut California in half and you're still left with two of five largest remaining states. You're also left with two distinct coronavirus outbreaks —

Next

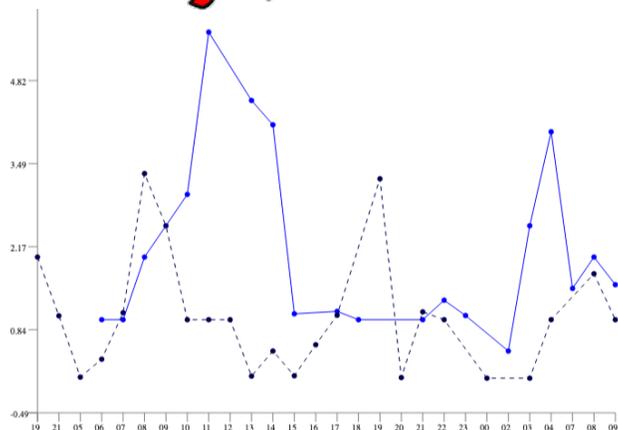
- [Blog](#) - [Privacy](#) - [Terms](#) - [YioopBot](#) - [Developed at SeekQuarry](#) -
 (c) 2020 Yioop - [PHP Search Engine](#)

 Type what to find



Trending Terms

Top Hourly		Top Daily		Top Weekly		Top Monthly	
Term	Score	Term	Score	Term	Score	Term	Score
United States	3.00	COVID-19	55.86	COVID-19	402.75	COVID-19	
Postmaster General		Kamala Harris	27.23	Donald Trump	210.83	Covid-19	
17-year-old	2.00	Donald Trump	25.16	Covid-19	171.60	Donald Trump	
mail-in	2.00	mail-in	22.19	Kamala Harris	140.83	Windows 10	
grand jury	2.00	Covid-19	11.39	Windows 10	61.00	social media	
COVID-19	1.56	5-year-old	6.67	social media	58.36	mail-in	
the next day	1.00	United States	6.25	mail-in	49.76	civil rights	
music video	1.00	Postal Service	6.00	national security	39.33	Supreme Court	
comic book	1.00	Windows 10	6.00	Air India Express	39.04	national security	
Chrissy Teigen	1.00	public health	5.34	college football	32.00	the week of	
Electoral College	1.00	social media	5.05	Pac-12	30.50	Kamala Harris	
Gaten Matarazzo	1.00	Supreme Court	4.87	Incidents in the	28.00	public health	
Ex-FBI	1.00	in-person	4.78	Life of a Slave Girl	28.00	Attorney General	
State Police	1.00	police officer	4.11	United States	27.70	in-person	
Donald Trump	1.00	Mar-a-Lago	4.00	Postal Service	27.70	high-profile	
anti-government	0.25	million-year-old	4.00	the week of	26.00	Pac-12	
district attorney	0.25	vote-by-mail	4.00	executive order	25.86	Third-Party	
Chulalongkorn	0.25	25-year-old	4.00	unemployment	24.89	Gavin Newsom	
University		Three-hundred-	4.00	benefits	24.89	unemployment	
police brutality	0.20	and-ninety-nine	4.00	in-person	24.14	benefits	
Roadshow	0.00	stay-at-home	4.00	pro-democracy	22.19	police officer	
video	0.00	Pac-12	3.50	Supreme Court	19.85	agreed to	
proper	0.00	mental health	3.50	civil rights	19.81	stay-at-home	
brother	0.00	sci-fi	3.50	Mar-a-Lago	18.36	Air India Express	
handle	0.00	payroll tax	3.18	3-year-old	17.33	Silicon Valley	
in	0.00	general election	3.17	Silicon Valley	16.57	Silicon Valley	
Sport	0.00	al-Qaeda	3.11	payroll tax	16.56	San Jose	
capable	0.00	Islamic State	3.11	drive-in	15.00	Earthquakes	
surprisingly	0.00	tight end	3.00	23-year-old	14.67	Homeland	
Bronco	0.00	first-of-its-kind	3.00	mental health	14.57	Security	
Ford	0.00	drive-in	3.00	police officer	14.51	T-Mobile	
activities	0.00	superpowers-in-	3.00	public health	14.41	back-to-school	
group	0.00	a-pill	3.00	My Hero	14.00	college football	
umbrellas	0.00	Chrissy Teigen	3.00	Academia	14.00	Ocasio-Cortez	
coolers	0.00	Kanye West	3.00	Live-Action	14.00	Southern	
nomes	0.00	state of the art	3.00	leave of absence	13.43	California	
prohibitions	0.00	third parties	3.00	agreed to	12.31	executive order	
scofflaws	0.00	New York Daily	3.00	72-year-old	12.25	Prime Minister	
beach	0.00	News	3.00	United States	12.00	Incidents in the	
tough	0.00	Executive Order	3.00	presidential election	12.00	Life of a Slave Girl	
trail	0.00	directory-as-	3.00	Hunter Pence	12.00	17-year-old	
Face	0.00	a-service	3.00	15-year-old	10.57	Amber Heard	
fine	0.00	first day of school	3.00	Carrie Lam	10.06	Roger Stone	
remain	0.00	United States	3.00	Carbon Dioxide	10.00	Social Media	
sneak	0.00	Postmaster General	3.00	90-year-old	10.00	Silicon Valley	
caught	0.00	Cardi B	2.67	Black-ish	10.00	Bank	
						anti-Semitic	



Computed Fri, 14 Aug 2020 09:38:54 -0700

- [Blog](#) - [Privacy](#) - [Terms](#) - [YioopBot](#) - [Developed at SeekQuarry](#) -
(c) 2020 Yioop - [PHP Search Engine](#)

When Yioop crawls a page it adds one of the following meta words to the page media:text, media:image, or media:video. RSS (or Atom) feed sources that have been added to Media Sources under the [Search Sources](#) activity are downloaded from each hour. Each RSS item on such a downloaded pages has the meta word media:news added to it. A usual web search just takes the search terms provided to perform a search. An Images, Video, News search tacks on to the search terms, media:image or media:video, or media:news. Detection of images is done via mimetype at initial page download time. Size of image and auxiliary information is then either determined either directly or using [EXIF](#) or [XMP](#) data in the image. At this time a thumbnail is generated. When search results are presented it is this cached thumbnail that is shown. So image search does not leak information to third party sites. On any search results page with images, Yioop tries to group the images into a thumbnail strip. This is true of both normal and images search result pages. In the case of image search result pages, except for not-yet-downloaded pages, this results in almost all of the results being the thumbnail strip. Video page detection is detected using the video web scraper. This checks whether there is an [Open Graph](#) meta tag on the page indicating the page is a video page. The thumbnail and video duration are also determined by Open Graph information. Thumbnails are then downloaded as a separate process at crawl time in the fetcher, converted to data urls, and added to the information associated with a video. The format of News search results is somewhat different from usual search results. News search results can appear during a normal web search, in which case they will appear clustered together, with a leading link "News results for ...". No snippets will be shown for these links, but the original media source for the link will be displayed and the time at which the item first appeared will be displayed. On the News subsearch page, the underneath the link to the item, the complete RSS description of the new item is displayed. In both settings, it is possible to click on the media source name next to the news item link. This will take one to a page of search results listing all articles from that media source. For instance, if one were to click on the Yahoo News text above one would go to results for all Yahoo News articles. This is equivalent to doing a search on: media:news:Yahoo+News . If one clicks on the News subsearch, not having specified a query yet, then all stored news items in the current language will be displayed, roughly ranked by recentness. If one has RSS media sources which are set to be from different locales, then this will be taken into account on this blank query News page.

The **Trends** page is computed by the FeedUpdate job and the TrendingHighlights job of the



trends page can be horizontally scrolled to reveal longer term trends such as yearly trends. The links on this page on keywords take one to a search result page for that keyword. Clicking on the score link or checking the check boxes next to keywords and clicking the *Chart* button takes one to a chart page which graphs how the trending term has changed over time.

[Return to table of contents](#) .

Search Page Footer and Search Menu

As one can see from the image of the main search form shown previously, the footer of each search and search result page has several links. Blog takes one to the group feed of the built in PUBLIC group which is editable from the root account, Privacy takes one to the Yioop installations privacy policy, and Terms takes one to the Yioop installations terms of service. The YioopBot link takes one to a page describing the installation's web crawler. These static pages are all Wiki pages of the PUBLIC group and can be edited by the root account. On a search page when a user clicks of the page hamburger menu, the **Search Web Context Menu** is revealed:



The screenshot shows a hamburger menu icon on the left, followed by the Yioop logo. The menu is organized into several sections:

- Welcome**
 - [Signin](#)
 - [Create Account](#)
- Categories**
 - [Web](#)
 - [Images](#)
 - [Videos](#)
 - [News](#)
 - [Trends](#)
- Settings**
 - Time:** All (dropdown)
 - Language:** English (dropdown)
 - Links/Page** Continuous (dropdown)
 - Open Results in Tabs:**
 - Safe Search:**
 - Save button
- Tools**
 - [Wiki Pages](#)
 - [Suggest A URL](#)

If a user is logged in, the top of this menu a link to the **Account Admin Context Menu** . If the person is not logged in, a Sign-In link and a Create Account link are shown at the top. Beneath this, the menu has links to te currently available subsearches. These default to Web, Images, Videos, News, and Trends. This is followed by a settings form that can be used to customize the current search. For example, one can control whether links open in new tabs or not or whether the results try to be family friendly only (safe) or not. If one is in an Image or Video subsearch, one can choose additional setting like the size of the image or duration of video. Finally, at the bottom of the Search Menu Context are links to a Suggest Url page and a link to the Public Wiki pages. Wiki's re described more under [Feeds and Wikis](#). The Suggest Url is shown below and allows a user to suggest links for the current or future crawls.



Suggest a site for the next web crawl. Up to ten sites per day can be accepted.

URL:

W t z v o j

Human Check:

Submit

▪ [Return to Yioop](#)

This link only appears if under Server Settings, Account Registration is not set to Disable registration.

[Return to table of contents](#) .

Search Operators

Turning now to the topic of how to enter a query in Yioop: A basic query to the Yioop search form is typically a sequence of words separated by whitespace. This will cause Yioop to compute a "conjunctive query", it will look up only those documents which contain all of the terms listed. Yioop also supports a variety of other search box commands and query types:

- **#num#** in a query are treated as query presentation markers. When a query is first parsed, it is split into columns based with **#num#** as the column boundary. For example, *bob #2# bob sally #3# sally #1#*. A given column is used to present **#num#** results, where **#num#** is what is between the hash marks immediately after it. So in the query above, the subquery *bob* is used for the first two search results, then the subquery *bob sally* is used for the next three results, finally the last column is always used for any remaining results. In this case, the subquery *sally* would be used for all remaining results even though its **#num#** is 1. If a query does not have any **#num#**'s it is assumed that it has only one column.
- Separating query terms with a vertical bar | results in a disjunctive query. These are parsed for after the presentation markers above. So a search on: *Chris | Pollett* would return pages that have either the word *Chris* or the word *Pollett* or both.
- Putting the query in quotes, for example "Chris Pollett", will cause Yioop to perform an exact match search. Yioop in this case would only return documents that have the string "Chris Pollett" rather than just the words Chris and Pollett possibly not next to each other in the document. Also, using the quote syntax, you can perform searches such as "Chris * Homepage" which would return documents which have the word Chris followed by some text followed by the word Homepage.
- If the query has at least one word not prefixed by -, then adding a `-' in front of a word in a query means search for results not containing that term. So a search on: of *-the* would return results containing the word "of" but not containing the word "the".
- Searches of the forms: **related:url** , **cache:url** , **link:url** , **ip:ip_address** are equivalent to having clicked on the Similar, Cached, InLinks, IP address links, respectively, on a summary with that url and ip address.

The remaining query types we list in alphabetical order:

chart

some_category:some_time_period:keyword_1:....:keyword_n : Draws a chart plotting the trending scores of keyword_1, ..., keyword_n for the search source category some_category for the time period some_time_period. For example, **chart:news:d:COVID-19:Donald_Trump** would plot news category keywords COVID-19 versus Donald Trump for the last day. some_time_period should be one of d (day), w



For example, `date:2011-01` would return all summaries where the response was a page not Found error.

date:Y, date:Y-m, date:Y-m-d, date:Y-m-d-H, date:Y-m-d-H-i, date:Y-m-d-H-i-s
returns summaries of all documents crawled on the given date. For example, `date:2011-01` returns all document crawled in January, 2011. As one can see detail goes down to the second level, so one can have an idea about how frequently the crawler is hitting a given site at a given time.

duration:time_scale
returns videos of given duration. Possibilities are: `duration:one-minute-minus`, `duration:one-minute-plus`, `duration:five-minute-minus`, `duration:five-minute-plus`, `duration:ten-minute-minus`, `duration:ten-minute-plus`, `duration:fifteen-minute-minus`, `duration:fifteen-minute-plus`, `duration:half-hour-minus`, `duration:half-hour-plus`, `duration:one-hour-minus`, `duration:one-hour-plus`, `duration:two-hour-minus`, and `duration:two-hour-plus`.

dns:num_seconds
returns summaries of all documents whose DNS lookup time was between `num_seconds` and `num_seconds + 0.5 seconds`. For example, `dns:0.5`.

filetype:extension
returns summaries of all documents found with the given extension. So a search: `Chris Pollett filetype:pdf` would return all documents containing the words Chris and Pollett and with extension pdf.

host:all
returns summaries of all domain level pages (pages where the path was /).

index:timestamp or i:timestamp
causes the search to make use of the IndexArchive with the given timestamp. So a search like: `Chris Pollett i:1283121141 | Chris Pollett` take results from the index with timestamp 1283121141 for Chris Pollett and unions them with results for Chris Pollett in the default index

if:keyword!add_keywords_on_true!add_keywords_on_false
checks the current conjunctive query clause for "keyword"; if present, it adds "add_keywords_on_true" to the clause, else it adds the keywords "add_keywords_on_false". This meta word is typically used as part of a crawl mix. The else condition does not need to be present. As an example, `if:oracle!info:https://oracle.com!/site:none` might be added to a crawl mix so that if a query had the keyword oracle then the site `https://oracle.com/` would be returned by the given query clause. As part of a larger crawl mix this could be used to make oracle's homepage appear at the top of the query results. If you would like to inject multiple keywords then separate the keywords using plus rather than white space. For example, `if:corvette!fast+car`.

info:url
returns the summary in the Yioop index for the given url only. For example, one could type `info:http://www.yahoo.com/` or `info:www.yahoo.com` to get the summary for just the main Yahoo! page. This is useful for checking if a particular page is in the index.

lang:IETF_language_tag
returns summaries of all documents whose language can be determined to match the given language tag. For example, `lang:en-US`.

media:kind
returns summaries of all documents found of the given media kind. Currently, the text, image, news, and video are the four supported media kinds. So one can add to the search terms `media:image` to get only image results matching the query keywords. For image search there are several variants depending on the size of the image as well: `media:image-tracking`, for one pixel or very small images; `media:image-small`, for images of less than 100000 pixels; `media:image-medium`, for images of less than 400,000 pixels; and `media:image-large` for larger images.



original mix had spaces, for example, use mix then to use the mix you would need to replace the spaces with plusses, *m:cool+mix* .

modified:Y, modified:Y-M, modified:Y-M-D

returns summaries of all documents which were last modified on the given date. For example, *modified:2010-02* returns all document which were last modified in February, 2010.

no:some_command

is used to tell Yioop not to perform some default transformation of the search terms. For example, *no:guess* tells Yioop not to try to guess the semantics of the search before doing the search. This would mean for instance, that Yioop would not rewrite the query *yahoo.com* into *site:yahoo.com* . *no:guess* also means that Yioop will not try to guess the language of the query and so won't restrict the search results to that language, it also won't try to guess if this should a safe search or an unrestricted search. *no:network* tells Yioop to only return search results from the current machine and not to send the query to all machines in the Yioop instance. *no:cache* says to recompute the query and not to make use of file cache.

numlinks:some_number

returns summaries of all documents which had some_number of outgoing links. For example, *numlinks:5*.

os:operating_system

returns summaries of all documents served on servers using the given operating system. For example, *os:centos* , make sure to use lowercase.

path:path_component_of_url

returns summaries of all documents whose path component begins with path_component_of_url. For example, *path:/phpBB* would return all documents whose path started with phpBB, *path:/robots.txt* would return summaries for all robots.txt files.

raw:number

control whether or not Yioop tries to do deduplication on results and whether links and pages for the same url should be grouped. Any number greater than zero says don't do deduplication.

robot:user_agent_name

returns robots.txt pages that contained that user_agent_name (after lower casing). For example, *robot:yioopbot* would return all robots.txt pages explicitly having a rule for YioopBot.

safe:boolean_value

is used to provide "safe" or "unsafe" search results. Yioop has a crude, "hand-tuned", linear classifier for whether a site contains pornographic content. If one adds *safe:true* to a search, only those pages found which were deemed non-pornographic will be returned. Adding *safe:false* has the opposite effect.

server:web_server_name

returns summaries of all documents served on that kind of web server. For example, *server:apache* .

site:url, site:host, or site:domain

returns all of the summaries of pages found at that url, host, or domain. As an example, *site:http://prints.ucanbuyart.com/lithograph_art.html* , *site:http://prints.ucanbuyart.com/* , *site:prints.ucanbuyart.com* , *site:.ucanbuyart.com* , *site:ucanbuyart.com*, *site:com*, will all returns with decreasing specificity. To return all pages and links to pages in the Yioop index, you can do *site:any* . To return all pages (as opposed to pages and links to pages) listed in a Yioop index you can do *site:all* . *site:all* doesn't return any links, so you can't group links to urls and pages of that url together. If you want all sites where one has a page in the index as well as links to that site, than you can do *site:doc* .

size:num_bytes

returns summaries of all documents whose download size was between num_bytes and num_bytes + 5000. num_bytes must be a multiple of 5000. For example,



was between `num_seconds` and `num_seconds + 60 seconds`. For example, `time:10`.

trending
some_category : returns a trending keywords page for the `some_category` search source category. For example, `trending:news` word return the page with what keywords are trending for all search sources of category news.

version:version_number
returns summaries of all documents served on web servers with the given version number. For example, one might have a query `server:apache version:2.2.9`.

weight:some_number or w:some_number
has the effect of multiplying all score for this portion of a query by `some_number`. For example, `Chris Pollett | Chris Pollett site:wikipedia.org w:5` would multiply scores satisfying `Chris Pollett` and on `wikipedia.org` by 5 and union these with those satisfying `Chris Pollett`.

Although we didn't say it next to each query form above, if it makes sense, there is usually an *all* variant to a form. For example, `os:all` returns all documents from servers for which `os` information appeared in the headers.

Result Formats

In addition to using the search form interface to query Yioop, it is also possible to query Yioop and get results in Open Search RSS format. To do that you can either directly type a URL into your browser of the form:

```
https://my-yioop-instance-host/?f=rss&q=query+terms
```

Or you can write AJAX code that makes requests of URLs in this format. Although, there is no official Open Search JSON format, one can get a JSON object with the same structure as the RSS search results using a query to Yioop such as:

```
https://my-yioop-instance-host/?f=json&q=query+terms
```

Sometimes it is advantageous to get the json data back as an argument to a Javascript function -- so called, [JSONP Format](#). This can be accomplished by add a callback parameter to your json query:

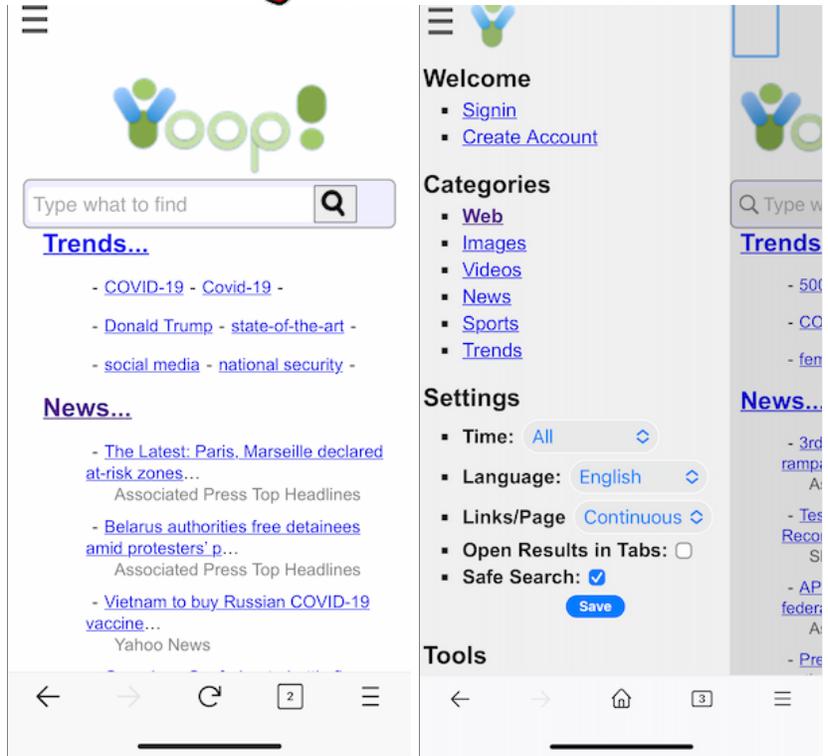
```
https://my-yioop-instance-host/?f=json&callback=myJavascriptFunction&q=query+terms
```

In addition, to changing the search format by adding the `f=format` to the query string, one can also add `&its=` the Unix timestamp of the search index you want to control the search index you are using, and `l=locale_tag`, such as `l=fr-FR`, to control the language of the search results.

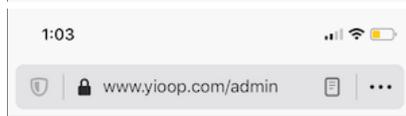
[Return to table of contents](#) .

Mobile Interface

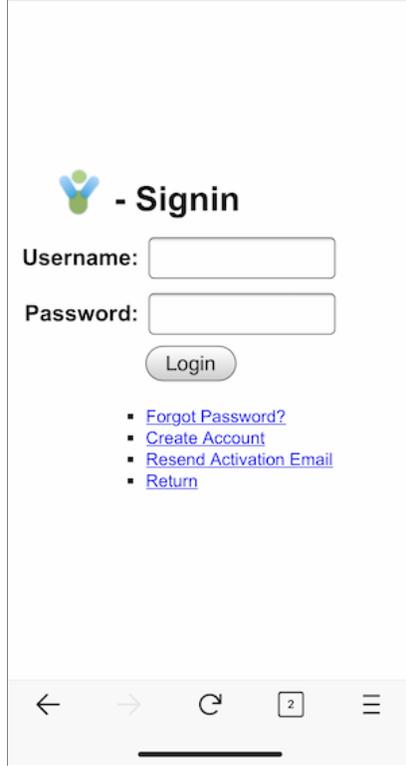
Yioop's user interface operates roughly the same on mobile devices as it does on desktop devices. Some scaling and modification of different interface components and icons as well as hiding of some unnecessary columns has been done to ensure the interface remains functional and responsive in a small screen setting. Below are some examples of how the search landing, search menu, and login pages look in Firefox on an iPhone:



The screenshot shows the yioop! search engine interface. On the left, the homepage features the yioop! logo, a search bar with the placeholder text "Type what to find", and sections for "Trends..." and "News...". The "Trends..." section lists items like "COVID-19 - Covid-19" and "Donald Trump - state-of-the-art". The "News..." section lists headlines such as "The Latest: Paris, Marseille declared at-risk zones..." and "Belarus authorities free detainees amid protesters' p...". On the right, a settings menu is open, showing options for "Welcome" (Signin, Create Account), "Categories" (Web, Images, Videos, News, Sports, Trends), "Settings" (Time: All, Language: English, Links/Page: Continuous, Open Results in Tabs: unchecked, Safe Search: checked), and "Tools".



The screenshot shows a mobile browser address bar with the URL "www.yioop.com/admin". The time is 1:03, and there are icons for signal strength, Wi-Fi, and battery level.



The screenshot shows the "Signin" page of the yioop! search engine. It features the yioop! logo and the title "- Signin". There are input fields for "Username:" and "Password:", followed by a "Login" button. Below the login button, there are links for "Forgot Password?", "Create Account", "Resend Activation Email", and "Return".

[Return to table of contents](#) .

User Accounts and Social Features

Registration and Signin

Clicking on the Sign In link from the Search Web, Group Discussion, or Group Wiki context

- Signin

Username:

Password:

Login

- [Forgot Password?](#)
- [Create Account](#)
- [Resend Activation Email](#)
- [Return](#)

Correctly, entering a username and password will then bring the user to the Account Activity portion of the Yioop website. The Account Admin Context hamburger menu shows what activities a user can do in this area. The default activity if Each Account page has a main panel where the current activity is displayed. When one first logs in this will default to the Manage Account activity. The Activity element allows the user to choose what is the current activity for the session. The choices available on the Account Admin Context menu depend on the roles the user has. A default installation of Yioop comes with two predefined roles Admin and User. If someone has the Admin role then the menu looks like:

Welcome, root

- [Manage Account](#)
- [Manage Users](#)
- [Manage Roles](#)
- [Sign Out](#)

Crawls

- [Manage Crawls](#)
- [Mix Crawls](#)
- [Manage Classifiers](#)
- [Page Options](#)
- [Results Editor](#)
- [Search Sources](#)
- [Web Scrapers](#)

Social

- [Combined Discussions](#)
- [My Groups](#)
- [Manage Groups](#)
- [Join Groups](#)

System Settings

- [Manage Machines](#)
- [Manage Locales](#)
- [Server Settings](#)
- [Security](#)
- [Appearance](#)
- [Configure](#)

On the other hand, if someone just has the User role, then their Activity element looks like:

[cpol.] 

Welcome, cpollett

- [Manage Account](#)
- [Sign Out](#)

Social

- [Combined Discussions](#)
- [My Groups](#)
- [Manage Groups](#)
- [Join Groups](#)

If under [Server Settings](#) , the **Monetization Type** is set to *Group Fees* , *Keywords Advertisements* , or *Group Fees and Keyword Ads* , then the Activity element will also contain:

Store

- [Manage Credits](#)
- [Manage Advertisements](#)

Over the next several sections we will discuss each of the Yioop account activities in turn.



By default, when a user first signs in to the Yloop admin panel the current activity is the Manage Account activity. This activity just lets user's change their account information using the form pictured below. It also has summary information about Groups and Feeds, Crawls and Indexes (Admin account only), and Crawl mixes. There are also helpful links from each of these sections to a related activity for managing them.

Welcome, root!

From this page you can access and control aspects of your account.

Account Details



Username:

First Name:

Last Name:

Email:

Groups and Feeds

Create or subscribe to groups to communicate with friends. You belong to 17 groups.

[Public \[Wiki|Manage\]](#) (49 posts, 49 threads, [Statistics](#))

Last Post: [Crawling Again](#)

[Yloop Software Help \[Wiki|Manage\]](#) (1285 posts, 283 threads, [Statistics](#))

Last Post: [-- Why the YIOOP Home Page does not display Trends, News and Sports in the default Search Engine Language ???](#)

[CS 154 Spring 2020 \[Wiki\]](#) (445 posts, 28 threads, [Statistics](#))

Last Post: [Approximating the smallest regular expression for a finite language.](#)

[CS 156 Fall 2014 \[Wiki\]](#) (142 posts, 75 threads, [Statistics](#))

Last Post: [Hw5 Scores Back!](#)

[\[See Combined Discussions\]](#) [\[See Your Groups\]](#) [\[Manage/Create Groups\]](#) [\[Join Groups\]](#)

Crawls and Indexes

Crawl and index the web or an existing archive and create a searchable index.

You have 1 active crawls, 3 previous crawl indexes.

[\[Manage Crawls and Indexes\]](#) [\[Search Query Statistics\]](#) [\[Manage Crawl Machines\]](#)

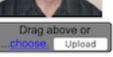
Crawl Mixes

Present search results from multiple indexes in the order you prefer. Share your search creations with groups of friends.

You have 5 crawl mixes.

[\[Create/Manage Crawl Mixes\]](#)

Initially, the Account Details fields are grayed out. To edit them, click the lock icon next to account details. This will allow a user to change information using their account password. A new user icon can either be selected by clicking the choose link underneath it, or by dragging and dropping an icon into the image area. The user's password must be entered correctly into the password field for changes to take effect when Save is clicked. Clicking the now opened Lock icon will cause the lock to close, these details to be grayed out, and not editable.

Last Name:
 Email:
 Language: 
 Bot User:
 Password:

The language drop down above can be used to select the language for the account, it will override the default language settings based on the language of the browser. The **Bot User** check box only appears if a Yioop site administrator has enabled Bot users as described in [Optional Server and Security Configurations](#) . When the **Bot User** check box is checked and saved, two additional fields **Bot Unique Token** and **Bot Url** will be displayed:

Account Details 



Username:
 First Name:
 Last Name:
 Email:
 Language: 
 Bot User:
 Bot Unique Token:
 Bot Callback URL:
 Password:

This url is used together with a Chat Bot story to allow a user to create automated responses to thread posts that contain `@the_name_of_the_bot_user` . Two chat bots, *Stock Bot* and *Weather Bot* , built using this mechanism can be found in the `src/examples` folder. The Chat Bot interface is described in the [Group Feed Chat Bots](#) section. If a user wants to change their password they can click the Password link label for the password field. This reveals the following additional form fields where the password can be changed:

Account Details 



Username:
 First Name:
 Last Name:
 Email:
 Language: 
 Bot User:
 Password:
 New Password:
 Retype:
 Recovery Answers: 
 
 

The **Recovery Answers** dropdowns are used to set the answers to the recovery questions you have for recovering your account in the event of a lost password. They may or may not be visible depending on Recovery Type chosen in the Security Activity.

[Return to table of contents](#) .

Managing Users, Roles, and Groups

The Manage Users, Manage Groups, and Manage Roles activities have similar looking forms as well as related functions. All three of these activities are available to accounts with the Admin role, but only Manage Groups is available to those with a standard User role. To describe these activities, let's start at the beginning... Users are people who have accounts to connect with a Yioop installation. Users, once logged in may engage in various Yioop activities such as Manage Crawls, Mix Crawls, and so on. A user is not directly assigned which activities they have permissions on. Instead, they derive their permissions



Username	First Name	Last Name	Email	Groups	Status	Actions
root	admin	admin	root@dev.null	3	Active	Edit Delete
test	test	test	chris@pollett.org	1	Active	Edit Delete
user0	First0	Last0	user0@email.net	1	Active	Edit Delete
user1	First1	Last1	user1@email.net	1	Active	Edit Delete
user10	First10	Last10	user10@email.net	1	Active	Edit Delete
user11	First11	Last11	user11@email.net	1	Active	Edit Delete
user12	First12	Last12	user12@email.net	1	Active	Edit Delete

The purpose of this activity is to allow an administrator to add, monitor and modify the accounts of users of a Yioop installation. At the top of the User list table is a dropdown used to control how many users to display at one time. If there are more than that many users, there will be arrow links to page through the user list. Clicking on the add icon, , reveals the Add User Form

Users  Row 0 to 30 of 30 Show 50  

Add User

Username:

First Name:

Last Name:

Email:

Status: Active 

Password:

Retype Password:

Username	First Name	Last Name	Email	Groups	Status	Actions
root	admin	admin	root@dev.null	3	Active	Edit Delete
test	test	test	chris@pollett.org	1	Active	 Edit Delete

The "Add User" form allows an administrator to add a new user to the Yioop system. Most of the fields on this form are self explanatory except the Status field which we will describe in a moment. At the top of the user list there is also a search link icon, , which can be used to bring up the following Search User form:

Users  Row 0 to 50 of 503 >> Show 50  

Search

Username Contains  No sort 

First Name Contains  No sort 

Last Name Contains  No sort 

Email Contains  No sort 

Status Equals  Any  No sort 

Username	First Name	Last Name	Email	Groups	Status	Actions
root	admin	admin	root@dev.null	3	Active	Edit Delete
test	test	test	chris@pollett.org	1	Active	 Edit Delete
weatherbot	Weather	Bot	chris@pollett.org	1	Active	 Edit Delete

This form can be used to find and sort various users out of the complete User List. If we look at the User List, the first four columns, Username, First Name, Last Name, and Email Address are pretty self-explanatory. The Groups column says how many groups the user belongs to. The Status column has a dropdown for each user row, this dropdown also



administrator, nor had the account activated by using an email link. Such a user can't create or post to groups or log in. On the other hand, such a user has reserved that username so that other people can't use it. A Banned user is a user who has been banned from logging, but might have groups or posts that the administrator wants to retain. Selecting a different dropdown value changes that user's status. Next to the Status column are two action columns which can be used to edit a user or to delete a user. Deleting a user, deletes their account, any groups that the user owns, and deletes any posts the user made. The Edit User form looks like:

[Add User Form](#)

User Information

Username:

First Name:

Last Name:

Email:

Status:

Roles:

Groups:

Password:

Retype Password:

User List Row 0 to 50 of 502 >> Show 50 [Search]

Username	First Name	Last Name	Email	Groups	Status	Actions
User0	First0	Last0	user0@email.ne t	2	Active	Edit Delete
User1	First1	Last1	user1@email.ne t	3	Active	Edit Delete

This form let's you modify some of the attributes of a users. There are also two links on it: one with the number of roles that a user has, the other with the number of groups that a user has. Here the word "role" means a set of activities. Clicking on one of these links brings up a paged listing of the particular roles/groups the user has/belongs to. It will also let you add or delete roles/groups. Adding a role to a user means that the user can do the set of activities that the role contains, adding a group to the user means the user can read that group, and if the privileges for non-owners allow posting then can also post or comment to that group's feed and edit the group's wiki. This completes the description of the Manage User Activity.

Roles are managed through the Manage Roles activity, which looks like:

Roles  Row 0 to 50 of 103 >> Show 50 

Name	Actions
Admin	Edit Delete
Bot User	Edit Delete
Role0	Edit Delete

Similar to the Manage User form, at the top of this activity, there is are add and search icons and controls, and beneath this a Role List. The controls of the Role List operate in much the same fashion as those of the User List described earlier. Clicking the add icon, opens a form to enter a role name. Entering a name and clicking *Save* , creates a blank role with that name in the Role List. We can then click *Edit* on this role in the Role List, to get a UI that allows one to add and delete Activities from this role:

Activities:

Name	Allowed Arguments	Actions
Manage Account	all	Delete
Feeds and Wikis	all	Delete
Manage Groups	all	Delete
Manage Locales	all	Delete
Add Activity Role		

Save

In the above, we have a Localizer role. We might have created this role, then used the Select Activity dropdown to add all the activities of the User role. A localizer is a person who can localize Yioop to a new language. So we might then want to use the Select dropdown to add Manage Locales to the list of activities. The Allowed Arguments field allows an administrator with a little more experience with the Yioop codebase to further customize to what extent a user can perform an activity. It defaults to **all**, which means no restrictions on the activity. Each Activity in Yioop has a set of allowed arguments. For example, for Manage Locales, these are "addlocale", "deletelocale", "editlocale", "editstrings", "search". If we wanted a localizer to only be allowed to edit strings of existing locales we might change the Allowed Arguments list to: editstrings, search. Search would allow this role to be able to search for particular locales to edit the strings for. Once we have created a role that we like, we can then assign user's that role and they will be able to perform all of the activities listed on it. If a user has more than one role, than they can perform an activity as long as it is listed in at least one role.

Groups are collections of users that have access to a group feed and a set of wiki pages. Groups are managed through the Manage Groups and Join Groups activities. The first of these looks like:

Subscribed Groups

Row 0 to 3 of 3 Show 

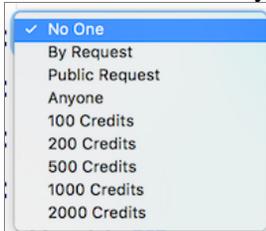
Name	Owner	Register	Access	Voting	Post Lifetime	Actions	
Help [Wiki Pages]	root [2 users] [Statistics]	Public Request	Read Write Wiki	+/- Voting	Never Expires	Edit	Delete
Public [Wiki Pages]	root [2 users] [Statistics]	Anyone	Read	No Voting	Never Expires	Edit	Delete
Search [Wiki Pages]	root [2 users] [Statistics]	Public Request	Read Write Wiki	+/- Voting	Never Expires	Edit	Delete

Unlike Manage Users and Manage Roles, the Manage Group and Join Groups activities belongs to the standard User role, allowing any user to create, manage, and join groups. The controls at the top of this table are similar in functionality to the controls we have already discussed for the User Lists table of Manage Users and the Roles List table of Manage Roles. This table let's a user manage their subscribed to groups, but does not let a user to see what groups other people have created and which might be joinable by the user. To see these a user can choose instead the Join Groups activity. For Subscribed groups, the statistics links only show to the owner of a group and have information about hourly, monthly, yearly group views, as well as hourly, month, and yearly statistics for threads and wiki pages. This information is generated only if the MediaUpdater process is running. As with the Manage Users and Manage Roles activities, clicking on the add icon bring up a form that allows a user to add a new group:

Register: 
 Access: 
 Voting: 
 Post Lifetime: 
 Encryption: 

The user who creates a group is set as the initial group owner.

For the above form, the **Register dropdown** says how other users are allowed to join the group: **No One** means no other user can join the group (you can still invite other users), **By Request** means that other users can request the group owner to join the group, but that the group is not publicly visible in the browsable group directory; **Public Request** is the same as By Request, but the group is publicly visible in the browsable group directory; and **Anyone** means all users are allowed to join the group and the group appears in the browseable directory of groups.



If under [Server Settings](#) 's under **Monetization Type** , either *Group Fees* or *Group Fees and Keyword Ads* is selected, then after **Anyone** , a user will also have the choice of charging various amounts of Yioop credits to join a group. If someone pays that amount of credits, then they could immediately join the group in question. It should be noted that the root account can always join and browse for any group. The root account can also always take over ownership of any group.

The **Access dropdown** controls how users who belong/subscribe to a group other than the owner can access that group. The possibilities are **No Read** means that non-members of the group cannot read the group feed or wiki, a non-owner member of the group can read but not write the group news feed and wiki; **Read** means that a non-member of the group can read the group news feed and the groups wiki page, but non-owners cannot write the feed or wiki; **Read Comment** means that a non-owner member of the group can read the group feed and wikis and can comment on any existing threads, but cannot start new ones, **Read Write** means that a non-owner member of the group can start new threads and comment on existing ones in the group feed, but cannot edit the group's wiki; finally, **Read Write Wiki** is wiki Read Write except a non-owner member can edit the group's wiki. The access to a group can be changed by the owner after a group is created. No Read and Read are often suitable if a group's owner wants to perform some kind of moderation. Read and Read Comment groups are often suitable if someone wants to use a Yioop Group as a blog. Read Write makes sense for a more traditional bulletin board.

The **Voting dropdown** controls to what degree users can vote on posts. **No Voting** means group feed posts cannot be voted on; **+ Voting** means that a post can be voted up but not down; and **+/- Voting** means a post can be voted up or down. Yioop restricts a user to at most one vote/post.

The **Post Lifetime dropdown** controls how long a group feed post is retained by the Yioop system before it is automatically deleted. The possible values are **Never Expires** , **One Hour** , **One Day** , or **One Month** .

The **Encryption dropdown** controls whether or not the posts to the group will be encrypted. Enabling encryption means that posts will no longer be searchable. Once you choose a group as encrypted, you are not able to change it to be unencrypted. Similarly,



is encrypted using the group key and a unique per post random salt vector. The idea is if an intruder steals only one of the two databases it will be difficult for them to decrypt the posts. Wiki pages and resources associated with a group are not encrypted regardless of the setting of this dropdown.

A default installation of Yioop has three built-in groups: **Public** , **Help** , and **Search** owned by root. Public has Read access and all users are automatically subscribed to it and cannot unsubscribe it. It is useful for general announcements and its wiki can be used as part of building a site for Yioop. The Help group's wiki is used to maintain all the wiki pages related to Yioop's integrated help system. When a user clicks on the help icon [?], the page that is presented in blue comes from this wiki. This group's registration is by default by Public Request and its access is Read Write Wiki. The **Search** group is used to maintain knowledge wiki callouts for search results. We saw an instance of this kind of callout when talking about the search result interface on the example query "zombie apocalypse" a callout appear in a panel on the right hand side of the results. This group has the same access permissions as the Help group. Unlike the other groups its wiki pages as typically edited using the **Results Editor** interface or are bulk populated using TokenTool.php's kwiki-seed command on a Wikipedia page count and Wikipedia dump file.

If on the Create/Join Group form, the name of a group entered already exists, but is not joinable, then an error message that the group's name is in use is displayed. If either anyone can join the group or the group can be joined by request, then that group will be added to the list of subscribed to groups. If membership is by request, then initially in the list of groups it will show up with access Request Join.

If instead of looking at already subscribed to groups, we select from the dropdown **Joinable Groups**, a list like the following is revealed:

Joinable Groups 							Row 0 to 50 of 117 >> Show 50  	
Name	Owner	Register	Access	Voting	Post Lifetime	Actions		
Bass [Wiki Pages]	anpollett [2 users] [Statistics]	Anyone	Read Write	+/- Voting	Never Expires	Edit	Join	
CS 116A Fall 2013 [Wiki Pages]	cpollett [1 users] [Statistics]	Public Request	Read	No Voting	Never Expires	Edit	Join	
CS 116B Spring 2014 [Wiki Pages]	cpollett [1 users] [Statistics]	Public Request	Read	No Voting	Never Expires	Edit	Join	
CS 146 Spring 2014 [Wiki Pages]	cpollett [1 users] [Statistics]	Public Request	Read	No Voting	Never Expires	Edit	Join	

If a group is joinable then the Join link in the Actions column of Joinable Groups table should be clickable. Let's briefly now consider the other columns of either the Subscribed or Joinable to Groups table. The Name column gives the name of the group. Group name are unique identifiers for a group on a Yioop system. In the Groups List table the name is clickable and takes you to the group feed for that group. The Owner column gives the username of the owner of the group and beneath this the number of members of the group. If you are the root account or if you are the owner of the group, then this field should be a clickable link that take you to the following form:

Transfer Group Owner

Name:

New Owner:

that can be used to transfer the ownership of a group. The next two column give the register and access information for the group. If you are the owner of the group these will be dropdowns allow you to change these settings. We have already explained what the Join link does in the actions column. Other links which can appear in the actions column



Edit Group

Name:

Register: 

Access: 

Voting: 

Post Lifetime: 

Encryption:

Feed: Import Discussions]"/>

Members: 32 users]"/>

The Register, Access, Voting, Post Lifetime dropdowns lets one modify the registration, group access, voting, and post lifetime properties for the group which we have already described before. Next to the Feed table header is a link to import discussions. Clicking this link reveals an upload form which can be used to upload RSS data for a feed discussion. Such data can be obtained by viewing the RSS pages for [phorum](#), [phpBB](#), or [Google Groups](#). Next to the Members table header is a link with the number of current members of the group. Clicking this link expands this area into a listing of users in the group as seen above. This allows one to change access of different members to the group, for example, approving a join request or banning a user. It also allows one to delete a member from a group. Beneath the user listing is a link which can take one to a form to invite more users.

Feeds and Wikis

The initial screen of the Combined Discussions page has an integrated list of all the recent posts to any groups to which a user subscribes:

Combined Discussions

10/08/2020

 [-- Hi there.](#) Bass
anpollett Yes

 [-- Hi there.](#) Bass (+0/-0) [\[Comment\]](#) [\[Edit\]](#) [\[X\]](#)
cpollett hi again

 [-- Hi there.](#) Bass
anpollett Thanks for joining

 [Hi there.](#) Bass (+0/-1) [\[Comment\]](#) [\[Edit\]](#) [\[X\]](#)
cpollett hey Andrew,
I joined your group!
Chris

 [-- anpollett joined Bass!](#) Bass
anpollett Hi there

 [-- Hi There Allan.](#) Chris Allan [\[Comment\]](#) [\[Edit\]](#) [\[X\]](#)
allanp73 hi back to you

This allows you to quickly see any new posts to any of your groups in one place. For each post, the post title link goes to the post in the thread to which it belongs. Post with titles beginning with "--" are follow up post to a post that started a thread. The gray group name links take you to the group the thread the post belongs to. In the example, above there are gray links to the Bass group and to the Chris Allan group. The green user name links (cpollett, anpollett, and allanp73 in the above) take you to a list of posts by that user. In the combined discussion view one can comment on existing posts, edit posts you created or to group for which you are the owner, and delete posts you wrote or in groups for which you are the owner. Next to the Yioop logo on pages related to Group Discussions and Wiki are two Grouping Icons:



Clicking the left icon is the same as clicking the My Groups link in the Account Admin Context, clicking the right link is the same as clicking the Combined Discussion link in the Account Admin Context when on a Group discussion page and is equivalent to clicking on the current group's wiki page list when in a Group Wiki content. The My Groups link takes one to an activity that looks like:

My Groups

↑↓ Filter Go

[CS 174 Spring 2021](#) [\[Wiki|Pages\]](#) (388 posts, 19 threads)
Last Post: [-- Practice Final Thread](#)

[CS 157B Spring 2020](#) [\[Wiki|Pages\]](#) (316 posts, 22 threads)
Last Post: [-- Practice Final Questions](#)

[CS 154 Spring 2020](#) [\[Wiki|Pages\]](#) (445 posts, 28 threads)
Last Post: [Approximating the smallest regular expression for a finite language.](#)

This activity presents information about each group a user belong to in terms of group activity rather than group capabilities/permissions. It also has a simple interface for searching for/sorting particular groups a user belong. Clicking on the link corresponding to the name of a group takes one to that group's discussion feed. Clicking on the *wiki* link for a group takes one to the Main page of that group's wiki. A single group discussion feed look like:



Start New Thread

12/07/2020

 [Why the YIOOP Home Page does not display Trends, News and Sports in the default Search Engine Language ???](#) (3 posts, 88 views) .
Last Post: 04/07/2020 - [root](#)

04/07/2020

 [Main Page Spiders / Image Keyword Crawl](#) (2 posts, 32 views) .
Last Post: 01/07/2020 - [root](#)

01/07/2020

 [Parse error: syntax error, unexpected '?'](#) (3 posts, 36 views) .
Last Post: 29/06/2020 - [archeerm](#)

22/06/2020

 [Yioop 7.0 Launch/Install Meeting](#) (1 posts, 118 views) .
Last Post: 22/06/2020 - [root](#)

Posts in the single group view are grouped by thread with the thread containing the most recent activity at the top. Notice next to each thread link there is a count of the number of posts to that thread. The content of the thread post is the content of the starting post to the thread, to see latter comments one has to click the thread link. There is now a Start New Thread button at the top of the single group feed as it is clear which group the thread will be started in. Clicking this button would reveal the following form:

Start New Thread

Subject

Post

B *I* U \$        

Drag items into the textarea to add them...[or click to select them.](#)

Save

Adding a Subject and Post to this form and clicking save would start a new thread. Posts can make use of [Yioop's Wiki Syntax](#) to achieve effects like bolding text, etc. The icons above the text area can be used to quickly add this mark-up to selected text. Although the icons are relatively standard, hovering over an icon will display a tooltip which should aid in figuring out what it does. Beneath the text area is a dark gray area with instructions on how to add resources to a page such as images, videos, or other documents. For images and videos these will appear embedded in the text of the post when it is saved, for other media a link to the resource will appear when the source is saved. The size allowed for uploaded media is determined by your php instances php.ini configuration file's values for post_max_size and upload_max_filesize. Yioop uses the value of the constant MAX_VIDEO_CONVERT_SIZE set in a configs/LocalConfig.php or from configs/Config.php to determine if a video should be automatically converted to the two web friendly formats mp4 and webm. This conversion only happens if, in addition, [FFMPEG](#) has been installed and the path to it has been given as a constant FFMPEG in either a configs/LocalConfig.php or in configs/Config.php.

Clicking on an existing thread link reveals posts for that thread:

 [Approximating the smallest regular expression for a finite language.](#) CS 154 Spring 2020 (+0/0) [Edit] [X]
cpollett
Hey Everyone,
The semester's over but I thought I'd post a link to a recent paper on approximating the smallest regular expression for a finite language:
<https://eccc.weizmann.ac.il/report/2020/079/>
Have a great summer!
Chris

10/05/2020

 [-- cant install yioop v6 on 1and1 host.](#) Yioop Software Help (+0/0) [Edit] [X]
cpollett
Hi Lisa,
.htaccess might be an issue. If the .htaccess file is not being processed because your hosting service doesn't allow it, then you will probably need to point your browser at the src folder to get Yioop to load the css correctly. I.e.,

some_url

would need to be

some_url/src

if the .htaccess file cannot be enabled.
Best, Chris

 [-- Crawl doesn't work.](#) Yioop Software Help (+0/0) [Edit] [X]

When a user starts a new thread or comments to an existing thread, several people will be notified via email provided they have their email address configured properly. For a new thread, the groups owner will be notified if they themselves weren't the ones starting the thread. If it was in fact the group's owner that starts a thread, then everyone who belongs to the group will be notified. For a comment to a thread, excluding the commenter, the group's owner, the person who started the thread, and anyone else who has commented on the thread would be notified via email. Note for a comment by the group's owner, it is still only the people participating in the thread who are notified.

The hamburger menu when opened in the Group Discussion portion of a Yioop instance can look like:

 <<Admin Menu

My Feed Places

- [Combined Discussions](#)
- [My Groups](#)

Recent Threads

- [Hi there](#)
- [Jan 29 In-Class Exercise](#)
- [Hi There Allan](#)
- [Least favorite F.R.I.E.N.D.S character](#)

Recent Groups

- [Bass](#)
- [CS 154 Spring 2020](#)
- [Chris Allan](#)
- [Pollett](#)

Bass Group Places

- [Bass Feed](#)
- [Bass Wiki](#)
- [Combined Discussions](#)
- [My Groups](#)

Recent Threads

- [Hi there](#)
- [Jan 29 In-Class Exercise](#)
- [Hi There Allan](#)
- [Least favorite F.R.I.E.N.D.S character](#)

Recent Groups

- [CS 154 Spring 2020](#)
- [Chris Allan](#)
- [Pollett](#)

 [X]

<<Admin Menu

User Places

- [User: cpollett](#)
- [Combined Discussions](#)
- [My Groups](#)

Recent Threads

- [May 6 In-Class Exercise](#)
- [setup problems](#)
- [SubsearchesWiki网页创建了!](#)

Recent Groups

- [CS 157B Spring 2020](#)
- [Yoop Software Help](#)
- [Public](#)
- [CS 154 Spring 2020](#)

The menu on the left is an example menu if one were on a combined discussion page, the one in the middle might be from a single group or thread page, and the one on the right comes from a User Feed. All have a gray link at the top that allows the user to see the Account Admin context menu. The places group of links that follow this gray link are useful links associated with the current, combined, group, thread, or user context. This is then followed with a collection of recent thread and group links.

Single Group, Threads, and User feeds of groups which anyone can join (i.e., public groups) all have RSS feeds which could be used in a news aggregator or crawled by Yioop. These feeds are discoverable to a news aggregator because a <link rel="alternate" ... > tag

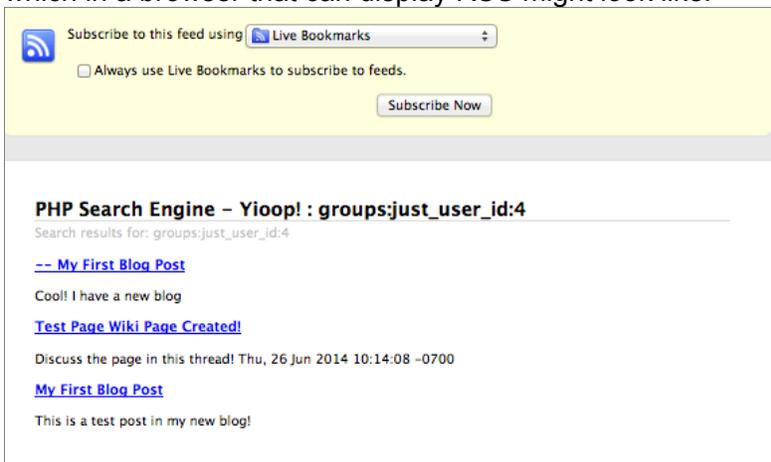
For example,

<https://www.yoop.com/user/4>

has as its RSS feed:

<https://www.yoop.com/user/4?f=rss>

which in a browser that can display RSS might look like:



Adding `f=json` and `f=json&callback=myFunction` also, work if JSON or JSONP formats are preferred.

There are several ways to get to a group's wiki pages. You can use the wiki links from either the Manage Groups or My Groups activity, or as we can see from the above there is a link to a group's Main wiki page from its Group Discussion Context Menu. Clicking on this link takes us to the group's main wiki page. If this has not yet been created this displays a page that looks like:



This page might be slightly different depending on whether the user has write access to the given group. The [Wiki Syntax Guide](#) link in the above takes one to a page that describes how to write wiki pages. On a wiki page, the hamburger menu take one to the Group Wiki Context Menu:

Main@Chris Allan

- [View](#)
- [Edit](#)
- [History](#)
- [Chris Allan Group Page List](#)
- [Chris Allan Feed](#)

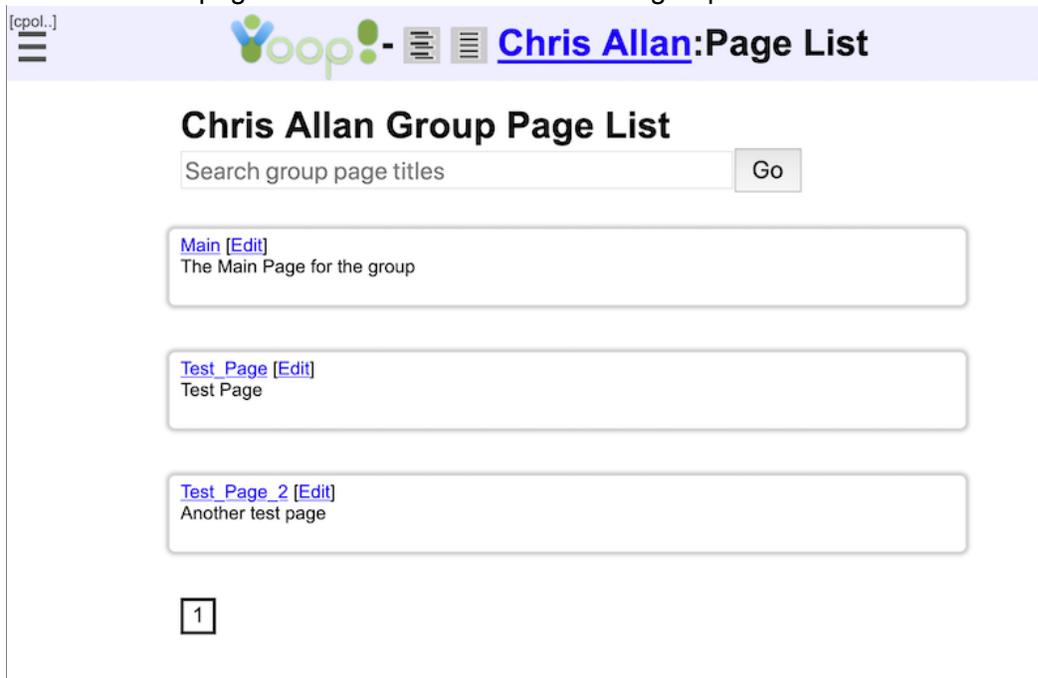
Recent Pages

- [Defenses@Pollett Masters Students](#)

Recent Groups

- [Bass](#)
- [CS 154 Spring 2020](#)
- [Pollett](#)

Notice besides editing a page there is an item link to read the page, there is an an item link to view the history of page, there may be an item link to see all the pages that share a link relationship with the page (only appears if any relationships to this page have been set up), there is an item link to discussion thread about the page, there is an item link to a page list of all pages in the group, there is an item link to the group's discussion feed, and there is an item link to the group's discussion feed. Beneath this there are links to recent discussion feeds and wiki pages the user has visited. The Pages link takes us to a screen where we can see all the pages that have been created for a group:



[cpol.]  -   [Chris Allan:Page List](#)

Chris Allan Group Page List

Search group page titles

[Main \[Edit\]](#)
The Main Page for the group

[Test_Page \[Edit\]](#)
Test Page

[Test_Page_2 \[Edit\]](#)
Another test page

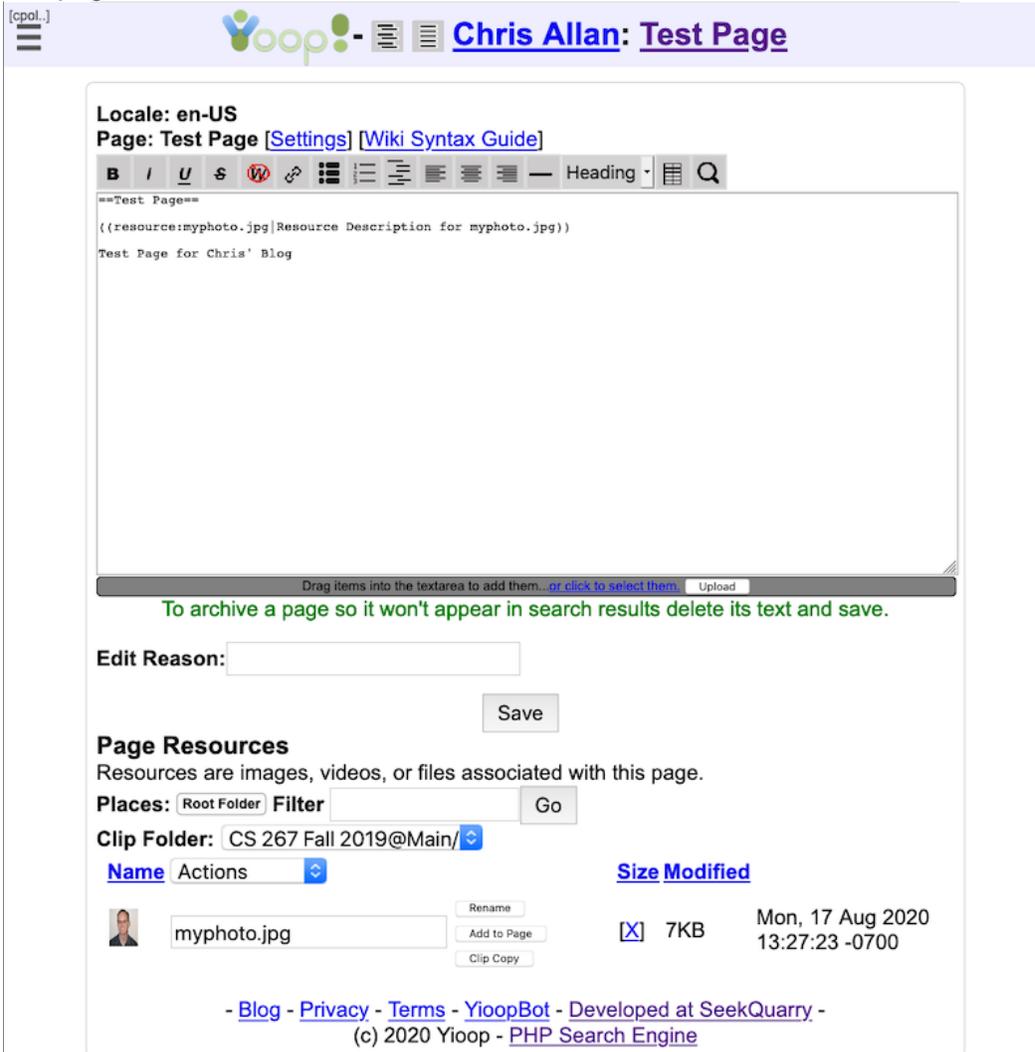
1

The search bar can be used to search within the titles of wiki pages of this group for a particular page. Suppose now we clicked on Test Page in the above, then we would go to that page initially in Read view:



Test Page for Chris' Blog

If we have write access, and we click the Edit link for this page, we work see the following edit page form:



The screenshot shows the edit interface for a page titled "Chris Allan: Test Page". At the top, there's a header with the Yioop logo and the page title. Below that, the locale is set to "en-US" and the page name is "Test Page". A rich text editor toolbar is visible, followed by a text area containing Wiki markup: `==Test Page==` and `{{resource:myphoto.jpg|Resource Description for myphoto.jpg}}`. Below the text area is a "Save" button and an "Edit Reason:" field. Underneath is the "Page Resources" section, which lists a resource named "myphoto.jpg" with a thumbnail, a size of 7KB, and a modification date of "Mon, 17 Aug 2020 13:27:23 -0700". At the bottom, there are navigation links for "Blog", "Privacy", "Terms", and "YioopBot", along with a copyright notice for 2020 Yioop - PHP Search Engine.

This page is written using Wiki mark-up whose syntax which as we mentioned above can be found in the [Yioop Wiki Syntax Guide](#) . So for example, the heading at the top of the page is written as

```
=Test Page=
```

in this mark-up. The buttons above the textarea can help you insert the mark-up you need without having to remember it. Also, as mentioned above the dark gray area below the textarea describes how to associate images, video, and other media to the document. Unlike with posts, a complete list of currently associated media can be found at the bottom of the document under the **Page Resources** heading. Links to Rename, Add a resource to the page, and Delete each resource can also be found here. Clicking on the icon next to a resource let's you look at the resource on a page by itself. This icon will be a thumbnail of the resource for images and videos. In the case of videos, the thumbnail is only generated

[\(First | Second\)](#) [2020-08-17T13:29:30-07:00](#). 1597696170. Edited by cpollett.(294 bytes). [\[Revert\]](#).

[\(First | Second\)](#) [2020-08-17T13:27:23-07:00](#). 1597696043. Edited by cpollett.(293 bytes). [\[Revert\]](#).

[\(First | Second\)](#) [2020-08-17T12:47:53-07:00](#). 1597693673. Edited by cpollett.(194 bytes). [\[Revert\]](#).

1

The Revert links on this history page can be used to change the current wiki page to a prior version. The time link for each version can be clicked to view that prior version without reverting. The First and Second links next to a version can be used to set either the first field or second field at the top of the history page which is labelled Difference: . Clicking the Go button for the Difference form computes the change set between two selected versions of a wiki document. This might look like:

[cpol...]

-  
Chris Allan: Test Page

Test_Page line differences between 2020-08-17T13:29:30-07:00 and 2020-08-17T13:27:23-07:00. [Back](#)

```

+++ Test_Page "1597696170"
--- Test_Page "1597696043"
-Test Page for Chris' Blog
+Test Page for Chris' Blog.
      
```

[- Blog](#) - [Privacy](#) - [Terms](#) - [YioopBot](#) - [Developed at SeekQuarry](#) -
 (c) 2020 Yioop - [PHP Search Engine](#)

This completes the description of group feeds and wiki pages.

[Return to table of contents](#) .

Group Feed Chat Bots

Yioop supports the creation of Chat Bots provided the Yioop administrator has [enabled bot users under Server Setting](#) . Chat Bots are automated users which can respond to posts in a group. Any Yioop user can be turned into a Chat Bot User by checking the [Is Bot User checkbox](#) in the Manage Accounts Activity. This enables the **Bot Story** activity for that user in the Account Admin Context Menu:

Chat Bot

- [Bot Story](#)

A chat bot user can be invited or join a group in the same way that a normal user [joins or is invited to a group](#) . Once a member of a group, if someone posts to the group's feed a message containing `@username_of_chat_bot` , Yioop will check the Chat Bot User's Bot Story to see if it can respond to the post, and if so, get it response:

4:44 pm

 **Bot Test.** [Edit] [X]
 root @weatherbot What is the weather in San Jose?

 **weatherbot** [Edit] [X]
 The weather is 89.56 with smoke in San Jose.

Comment

The Chat Bot story for a Yioop Chat Bot is controlled under the **Bot Story** activity:

 [Admin](#) [Bot Story]

Bot Story Row 0 to 2 of 2 Show 50

Add Pattern ?

Request Expression:

Trigger State:

Remote Message:

Result State:

Response:

Pattern	Actions
<i>Request Expression:</i> What is your name? <i>Trigger State:</i> 0 <i>Remote Message:</i> <i>Result State:</i> 0 <i>Response:</i> Weatherbot	Edit Delete
<i>Request Expression:</i> What is the weather in \$location? <i>Trigger State:</i> 0 <i>Remote Message:</i> getWeather,\$location <i>Result State:</i> 0 <i>Response:</i> \$REMOTE_RESPONSE	Edit Delete

[- Blog](#) - [- Privacy](#) - [- Terms](#) - [- ThisSiteBot](#) - [- Developed at SeekQuarry](#) -
 (c) This Site - [- This Search Engine](#)

A **Bot Story** is a set of patterns that control how a chat bot reacts to group threads posts for groups to which it belongs. As we can see above, the activity consists of the particularly pattern that is about to be added or is currently being edited and under this a table of currently added patterns. Using the **Edit** links a previously entered pattern can be edited; using the **Delete** links it can be deleted. A bot story pattern consists of the following components which are configurable in the Bot Story activity:

Request Expression

This and the trigger state are used to determine if a bot will react to a post. If a post contains @name_of_the_bot followed by some phrase or phrases which match the expression and the trigger state matches the trigger state of the bot for that user, then the pattern will apply. Request expressions are allowed to contain variables. They are strings beginning with \$ followed by a sequence of word characters. For example, the expression:

What is the weather in \$location?

would match the string

What is the weather in San Jose?

and the value of \$location would get bound to San Jose in this match.

Trigger State



expression for the pattern when the pattern is activated. In the add pattern/edit pattern forms one can use simple strings or strings containing variables in defining a trigger state. For example, "0", "asleep" are simple trigger states. One can also have "1\$location". If the request expression for a pattern was "What is the weather in \$location?" and the user was in state "1San Jose" and posted a message "What is the weather in San Jose?" then this pattern would activate.

Remote Message

If a bot url has been configured for a chat bot, then when a pattern is activated a request will be made to that url as part of computing the response the chat bot makes to the message which was just posted. The url request will have as part of its query string a variable *remote_message* which comes from this field of the Bot Pattern. The Remote Message can be any string and is allowed to have variables in it. So for example, a pattern's Remote Message might be `getWeather,$location`. When the value of `$location` is substituted with might become `getWeather,San Jose`. This tells the bot url site what action to perform with what value.

Result State

This is the state the chat bot should enter for that user after the pattern is applied. It is allowed to be an arbitrary string and can have variables in it. These will be interpolated when the pattern is applied.

Response

This is the text that the chat bot will post back to the thread in question in response to a user request. It is a string and can have variables. In addition, to variables bound from the Request Expression, it can make use of `$REMOTE_RESPONSE` which has the string returned from the Bot url call (if there was one), and it can make use of `$USER_NAME`, the name of the user that the post was in response to.

[Return to table of contents](#) .

Keyword Advertising

Keyword advertising allows a user with the Business Role to pay to associate a query term or phrase with a text-based advertisement for a fixed number of days. By default, Yioop does not come with an advertisement platform turned on. To enable keyword advertising, one can select **Keyword Advertisements** from the dropdown in the **Ad Server Configuration** fieldset of the [Server Settings](#) activity. This turns on keyword advertising. A user can then choose to upgrade their account to do advertising as described in the [Manage Account](#) section. If this is done, the following new activities will appear in the Account Admin Context menu:

Store

- [Manage Credits](#)
- [Manage Advertisements](#)

The **Manage Credits** activity allows a user to purchase Ad Credits with a credit card. Ad Credits in turn can be used to pay for advertisements created in **Manage Advertisements** . The **Manage Credits** activity looks like

Balance: 940 credits

Credit Transactions 

Row 0 to 3 of 3 Show 50  

Purchase Credits 

Quantity: 

Credit Card Number:

CVC:

Expiration:  / 

Using the Purchase button charges the above card the Quantity field's amount in US dollars and agrees to the [Program Terms](#).

Type	Amount	Date	Total
Starting Balance	0	Mon, 17 Aug 2020 17:00:19 -0700	0
Buy Credits	1000	Mon, 17 Aug 2020 17:01:22 -0700	1000
Buy Ad	-60	Mon, 17 Aug 2020 17:21:48 -0700	940

The **Purchase Ad Credits** form is used to actually purchase ad credits. On it, the **Quantity** dropdown specifies the number of credits one wants to purchase at what price. The **Card Number** field should be filled in with a valid credit card, in the **CVC** field you should put the three or four digit card verification number for your card, finally, the **Expiration** dropdown is used to set your cards expiration date.

Beneath the Purchase form is the list of **Ad Credit Transactions** that have been made with your account.

Clicking on the **Manage Advertisements** activity link will display a form like:

Ad Body : Summer Slush is here!

Destination URL:

Campaign Duration: 
Start day is day of purchase.

Keywords:

We split the remainder of this section into three subsections: the Manage Advertisements activity, the mechanics of how ad bids are calculated, and how payment processing is done.

The Manage Advertisements Activity

Returning to the screenshot at the end of the last section, the **Ad Title** , **Ad Body** , **Destination URL** fields can be used to create a text-based advertisement. What this ad will look like appears in the **Preview** area.

The **Duration** dropdown controls how many days the ad campaign will run for. The campaign starts on the date of purchase. The first day of a campaign starts from the time of purchase and last until midnight in the Yioop instance's timezone. Subsequent days start at midnight and last till the following midnight.

Keywords should consist of a comma separated list of words or phrases. Each word or phrase has a minimum bid for each day based on demand for that keyword. If no one so far has purchased an ad for any of the keywords, then this minimum is \$1/day/(word or phrase). Otherwise, it is calculated using the total of the bids so far. Keywords can include Yioop meta words such as media:news.

The **Calculate Bid** button computes the minimum cost for the campaign you have chosen, and then presents a form to receive your credit card information.

Ad Body :

Destination URL:

Campaign Duration:
Start day is day of purchase.

Keywords:
Preview:  [Best Lemonade](#)
 Summer Slush is here!

Minimum Bid Required:

Expensive word

Budget:
Using the Purchase button deducts the Budget field amount from your 1000 available credits. [Buy more ad credits.](#)

On this form the static field **Minimum Bid Required** displays the minimum amount required to pay for the advertisement campaign in question. The **Expensive word** static field says for your campaign which term contributes the most to this minimum bid cost. The Budget fields allows you to enter an amount greater than or equal to the minimum bid that you are willing to pay your ad campaign. If there have been no other bids on your keywords then the minimum bid will show your ad 100% of the time any of your keywords are searched for on the Yioop instance. If, however, there have been other bids, your bid amount as a fraction of the total bid amount for that day for the search keyword is used to select a frequency with which your ad is displayed. Thus, it can make sense to bid more than the minimum required amount.

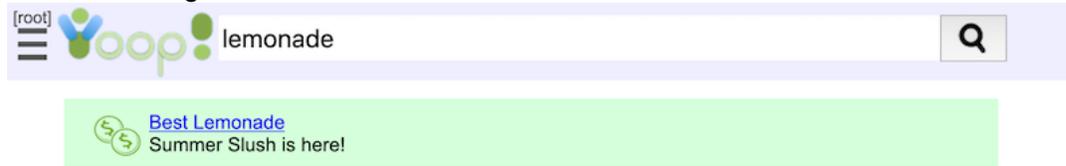
If you need to edit the keywords or other details of your ad before purchasing it, you can click the **Edit Ad** button; otherwise, clicking the **Purchase** button completes the purchase of your Ad campaign. After purchase, your ad will appear in the Advertisement List.

Advertisement List  Row 0 to 1 of 1 Show 50  

Ad Name	Username	Keywords	Budget	Date	Views/Clicks	Status	Actions
Best Lemonade	root	LEMONADE,SOFT DRINK	60	2020-08-17 / 2020-09-15	0 / 0	Active	Edit Deactivate

The **Advertisement List** is a list of all current and previous ads in descending order by when the campaign was created. This form appears slightly differently for an administrative user than a normal user. For the former, as in the above image, there is a *Username* column, and ads not belonging to the current user will be shown. For these ads, the administrator gets a link to suspend the ad. In the normal user mode, the table does not have a *Username* column and only the current user's ads are displayed. A normal user can deactivate and reactivate their own ads, but can't reactivate an ad that was suspended by an administrator. A user can use the **Edit** button next to an ad to change the text or links associated with an active ad. One can determine how many times the ad was served and how many times an ad was clicked so far during a campaign by looking at the

It has an opportunity to be displayed at the top of the search results. This would typically look something like:



Bid Mechanics

In this subsection, we describe how the minimum purchase price for an ad is determined, how ads are chosen for display, and try to give some intuitions on the ad bidding process. The minimum purchase of an ad campaign is determined by adding together the minimum costs of the different keywords involved in that campaign. Let CR denote the value of an ad credit. By default in Yioop, one ad credit is about 1 cent. As an example, if one had three keywords and the cost of each separately was 7CR, 10CR, and 5CR, then the minimum purchase price would be 22CR. In turn, the minimum purchase price of a keyword for an ad campaign, is the sum of the current costs of the ad for each day during the campaign. For example, suppose we were running an ad on the single keyword *lemonade* for one week, the minimum cost for the current day was 10CR, and for subsequent days was 9CR, 7CR, 9CR, 5CR, 2CR, 1CR. Then the minimum purchase price for that keyword for the seven days would be $(10 + 9 + 7 + 9 + 5 + 2 + 1)CR = 43CR$. The minimum cost of a keyword on a particular day is half the sum of the purchase for that keyword for that day so far rounded up. For example, suppose for a particular day, ads on the word *lemonade* had been purchased for 1CR, 1CR, 1CR, 2CR, 3CR, and 4CR. The total credit amount spent on *lemonade* for that day would then be 12CR. So the minimum next bid for the word *lemonade* for that day would be half that amount round up, in this case, 6CR.

The Yioop keyword advertising system allows one to bid higher than the minimum purchase price for an ad campaign. As we will see below there might be strategic reasons for doing

this. Let $\alpha = \frac{\text{bid amount}}{\text{minimum bid amount}}$. Then the bid amount this corresponds to for a keyword for a particular day, is α times the minimum bid for that keyword for that day. For example, suppose the minimum purchase price for a seven day campaign on the keywords *lemonade* and *soft drink* was 24CR, but the campaign was purchased for 36CR. Then α would be 1.5. If for *lemonade* the daily minimum bids were 4CR, 2CR, 2CR, 2CR, 2CR, 2CR, 2CR and those for *soft drink* were 2CR, 1CR, 1CR, 1CR, 1CR, 1CR, 1CR, then this ad purchase would correspond to bids for these days of 1.5 times this minimum bids for these days. For *lemonade*, this is relatively straightforward and results in the bids of 6CR, 3CR, 3CR, 3CR, 3CR, 3CR, 3CR. For *soft drink*, we might end up with fractional values. To handle this, Yioop rounds down the fractional value, but also keeps track of the sum of the left over fractions so far, if this sum ever exceeds 1, then 1CR is added to that days bid, and left over is deducted by one. So on the *soft drink* bids we get, $1.5 \times 2CR = 3CR$, $\lfloor 1.5 \times 1 \rfloor = 1CR$ with the left over at 0.5, $\lfloor 1.5 \times 1 \rfloor = 1CR$ with the left over now at $0.5 + 0.5 = 1$, so the bid value is upped to 2CR and the left over is deducted by 1 back to 0. Continuing in this fashion for the remaining bids, yields 1CR, 2CR, 1CR, 2CR. If, due to round off, there is still a left over amount after the last keyword on the last day, then an additional bid of 1 is made for this last keyword on the last day.

Once an ad is purchased, it can be displayed when a user performs a search. Ads are only displayed for simple queries, that is, queries not involving the operators | or #. Given such a query, Yioop performs a look up to see if there are any ads for that day which exactly match the query phrase. For example, if the query was "soft drink" then Yioop would do a look up to see if there are any active ad campaigns with "soft drink" as one of their keywords. If there are, Yioop moves on to an ad selection phase, if there are no such ads, then Yioop



then Yioop selects the term for which the most money has been spent and proceeds to the ad selection phase using that term.

In the ad selection phase, Yioop arranges the ad campaigns for the chosen keyword in the order in which the purchases were made. It uses a pseudo-random number generator (PRNG) to pick an integer less than the total bid on that keyword for the current day. It then computes sums of ad prices from initial sublists of campaigns until it finds the first partial sum greater than or equal to the integer. Finally, it chooses the ad campaign that corresponded to the last summand in this sublist. For example, suppose the query was "soft drink" and there were four purchased ads with the "soft drink" keyword for that day. Bid 1 was 1CR, Bid 2 was 1CR, Bid 3 was 3CR, and Bid 4 was 4CR. The total spend on this keyword for the current day would be 9CR. If the PRNG picked the number 3 from among the choices 0, 1, 2, 3, 4, 5, 6, 7, 8, then Yioop would start computing initial sums. The first value greater than this is 4. The blank sum 0CR is less than 4CR, so it would add Bid 1 to get the partial sum 1CR, then it adds Bid 2 to get the partial sum 2CR, and finally, it would add Bid 3's bid to get the partial sum 5CR. As $5CR \geq 4CR$, and the last bid added was Bid 3's, it would be Bid 3's ad that would be displayed for this particular search. As the PRNG chooses a different number reasonably uniformly from 0, 1, 2, 3, 4, 5, 6, 7, 8 each time the query "soft drink" is made, the frequency with which a given ad for that term is displayed will be proportional to the amount that that ad was purchased for.

We now conclude this section with some observations about this bidding system. The first observation is that days further in the future will tend to have fewer bids for keywords on those days as there has been less time for people to bid on those days. Currently, the maximum campaign length is 180 days, but most campaign will probably be shorter. Hence, on a cost/day basis for a minimum bid for a keyword is like to be lower for a longer campaign. A second observation is that if your valuation of a keyword on a given day is more than twice anyone else's valuation, then if you bid your valuation first for that day, no one else will try to bid as half the total will be more than their valuation. So truthful bidding can be advantageous.

Many keyword advertisement systems use some kind of auction system, so it is interesting to compare Yioop's system with an auction. One could imagine have an open, ascending price auction for a keyword to purchase all impressions/clicks that that ad generates from Yioop searches for a day. Suppose for the keyword lemonade, searches generate on average 100 impressions in a day. If two bidders both value an impression at 1CR, then both would value the lemonade keyword for one day at 100CR. The first person would bid 100CR, and the second person would not bid as then he would be paying more than a 1CR/impression. The first person wouldn't underbid as then the second would have the opportunity to take all 100 impressions. Now consider what would happen with Yioop's bidding system. If the first person bids 100CR for lemonade for the day, then the second person would have a minimum bid of 50CR. If the second person bids 50CR, then they will receive about $\frac{50}{100 + 50} \cdot 100 \approx 33$ impressions, so the cost/impression will be over a

credit. Higher bids would only make the cost/impression worse, so the second bidder would not bid. Hence, in this situation Yioop's bidding system has the same effect as an auction.

Now consider a situation in which one has two bidders, both of whom value an impression at 1CR, but both of whom have a limited budget of 50CR. In the auction system, the first bidder would bid 50CR, and the second bidder, although he might want to bid higher, doesn't have the wherewithal to do so. So the first bidder gets all the clicks for 0.5CR each, the auction house has lost potential revenue on the item, and the second bidder doesn't get there ad displayed at all. In the Yioop system, the second bidder could bid 50CR as well, each bidder would get half of the 100 impressions, that is, 50 each, and they would each be paying 1CR/impression. So in this situation the Yioop system is advantageous to the seller and gives more bidders an opportunity to have their ads displayed at a fair price.

Payment Processing

Payment processing entails actually charging the credit card based on the data collected



allows you to experiment with how the advertising platform works. The link below take you to a version of this file which uses [stripe.com](#) for payment processing:

[Yioop Keyword Ad Script.](#)

One advantage to using stripe.com for handling payments is that your never store the credit data on your Yioop instances servers -- the data is only sent to the stripe.com servers. You should still make sure the page that collect the purchase form is on uses https rather than http.

[Return to table of contents](#) .

Crawling and Customizing Results Performing and Managing Crawls

The Manage Crawl activity in Yioop looks like:

The screenshot shows the Yioop Admin interface for managing crawls. At the top, there's a navigation bar with the Yioop logo and an 'Admin [Manage Crawl]' link. Below this, the 'Crawl Status' section is active, showing details for a crawl named '1. Webscale 2020'. A 'Stop Crawl' button is visible. The status includes a search index link, timestamp, start time, channel, and various memory usage statistics for the indexer, scheduler, and fetcher. It also lists visited URLs and the most recent fetcher. Below the status, a list of 'Most Recent Urls' is shown with several links to Pinterest and other websites. At the bottom, a table lists active crawls, including 'feed' and 'Webscale 2020'. The table has columns for Description, Timestamp, Visited/Extracted Urls, and Actions. A footer contains links for Blog, Privacy, Terms, YioopBot, and Developed at SeekQuarry, along with a copyright notice for 2020 Yioop - PHP Search Engine.

Crawls in Yioop are performed using Queue server and Fetcher processes which run on server machines. Under the Crawl Status heading in the image above, we see a link to **Manage Machines** which allows us to configure these processes on the different machines that the Yioop instance is aware of. This link also appear in the Account Admin Context Menu. Beneath this link in the screenshot above, one can see there is a single currently active crawl, *Webscale 2020* . The number 1 by the *Webscale 2020* name hints about Yioop's ability to do multiple simultaneous crawls in which case one gets an ordered list of all the active crawls. The **Stop** button next to the crawl name, *Webscale 2020* , in this case, can be used to stop the given crawl. To stop larger scales can take some time as the indexes are merged, so one should avoid stopping processes under Manage Machines just after Stop a crawl until the crawl appear in the Crawls list further down the page. The line under the crawl name/ stop button line has up to two links: a potential [Search Index] link



crawl. Under the two links are some statistics about the ongoing crawl. Finally, under this is a heading **Crawls [Query Statistics]** beneath which is a table of previously completed crawls. The Query Statistic link takes one to a page where one can see statistics about what queries user of Yioop are performing:

Last Hour		Last Day		Last Month		Last Year	
	Views		Views		Views		
lang:default-major highlight:3	5	lang:default-major highlight:3	78	lang:default-major highlight:3	2629	lang:default-major	
tuto lang:fr	2	呂主席	8	lang:default-major highlight:1	107	lang:default-major highlight:3	
Go	2	lang:default-major highlight:1	6	x lang:fr	63	lang:default-major highlight:1	
Web search	2	do lang:fr	6	do lang:fr	63	COVID-19	
août lang:fr	1	COVID-19	5	60 lang:fr	54	lang:fr	
barcelone lang:fr	1	mac lang:fr	4	35 lang:fr	51		

Continuing with our Manage Crawls Activity screenshot example, clicking the add icon next **Crawls [Query Statistics]** reveals a form that would allow one to give a name of a new crawl and start it.

New Crawl

Description: [Options](#)

Next to the Start New Crawl button is an Options link, which allows one to set the parameters under which the crawl will execute. We will return to what the Options page looks like in a moment.

Prerequisites for Crawling

Before a new crawl can start, at least one instance of QueueServer.php script and at least one Fetcher.php script must be running. In a default Yioop configuration, a single machine with each of these scripts is configured under Manage Machines. When you click the **Start** button for a new crawl, Yioop checks if these scripts are already running, and if not, starts them for you and proceeds to conduct the crawl as per the crawls options you have chosen under Options. For debugging purposes, it is useful to know a little more about starting and stopping QueueServer.php and Fetcher.php, so we discuss this first. If you want to just move on to setting crawl options, please skip ahead to the [Specifying Crawl Options and Modifying Options of the Active Crawl](#) subsection. Running the QueueServer.php and Fetcher.php scripts can be done either via the command line or through a web interface. As described in the Requirements section you might need to do some additional initial set up if you want to take the web interface approach with a non default number of queue servers or fetchers. On the other hand, the command-line approach only works if you are using only one queue server. You can still have more than one fetcher, but the crawl speed in this case probably won't go faster after ten to twelve fetchers. Also, in the command-line approach the queue server and name server should be the same instance of Yioop. In the remainder of this section we describe how to start the QueueServer.php and Fetcher.php scripts via the command line; the GUI for Managing Machines and Servers section describes how to do it via a web interface. To begin open a command shell and cd into the executables subfolder of the Yioop folder. To start a queue server type:

```
php QueueServer.php terminal
```

To start a fetcher type:

```
php Fetcher.php terminal
```

The above lines are under the assumption that the path to php has been properly set in your PATH environment variable. If this is not the case, you would need to type the path to



the QueueServer.php script, log lines related to scheduling and will all have scheduler in them, of these those with the prefix FB say when the scheduler is preparing a batch of files for a Fetcher to download. Lines related to creating the search index will all have Indexer in them. Log lines containing the word idle, give you an estimate on the percentage of time the Queue Server is doing an active task rather than sleeping and waiting for data from Fetchers. Fetchers also have idle log lines and they log a line any time the fetcher downloads a URL with the name of the URL. Assuming that you have done the additional configuration mentioned above that are needed for the GUI approach managing these scripts, it is also possible to run the queue server and fetcher programs as daemons. To do this one could type respectively:

```
php QueueServer.php start
```

or

```
php Fetcher.php start
```

When run as a daemon, messages from these programs are written into log files in the WORK_DIRECTORY/log folder. To stop these daemons one types:

```
php QueueServer.php stop
```

or

```
php Fetcher.php stop
```

Once the Queue Server is running and at least one Fetcher is running, the Start New Crawl button should work to commence a crawl using them. It will up to a minute or so for information about a running crawl to show up in the Crawl Status fieldset. During a crawl, it is possible for a Fetcher or the Queue Server to crash. This usually occurs due to lack of memory for one of these programs. It also can sometimes happen for a Fetcher due to flakiness in multi-curl. If this occurs, simply restart the Fetcher in question and the crawl can continue. A queue server crash should be much rarer. If it occurs, all of the urls to crawl that reside in memory will be lost. To continue crawling, you need to restart the queue server and then typically resume the crawl through the web interface. If there are no unprocessed schedules for the given crawl (which usually means you haven't been crawling very long), it is not possible to resume the crawl. Now that we have described what is necessary to perform a crawl, we return to how to set the options for how the crawl is conducted.

Common Crawl and Search Configurations

When testing Yioop, it is quite common just to have one instance of the Fetcher and one instance of the Queue Server running, both on the same machine and same installation of Yioop. In this subsection we wish to briefly describe some other configurations which are possible and also some src/configs/Config.php configurations that can affect the crawl and search speed. The most obvious Config.php setting which can affect the crawl speed is NUM_MULTI_CURL_PAGES. A fetcher when performing downloads, opens this many simultaneous connections, gets the pages corresponding to them, processes them, then proceeds to download the next batch of NUM_MULTI_CURL_PAGES pages. Yioop uses the fact that there are gaps in this loop where no downloading is being done to ensure robots.txt Crawl-delay directives are being honored (a Crawl-delayed host will only be scheduled to at most one fetcher at a time). The downside of this is that your internet connection might not be used to its fullest ability to download pages. Thus, it can make sense rather than increasing NUM_MULTI_CURL_PAGES, to run multiple copies of the Yioop fetcher on a machine. To do this one can either install the Yioop software multiple times or give an instance number when one starts a fetcher. For example:

```
php Fetcher.php start 5
```

would start instance 5 of the fetcher program.

Once a crawl is complete, one can see its contents in the folder WORK_DIRECTORY/cache/IndexDataUNIX_TIMESTAMP. In the multi-queue server setting each queue server



server is storing ten million or so pages, the corresponding IndexDataUNIX_TIMESTAMP folder might be around 200 GB. Two main sub-folders of IndexDataUNIX_TIMESTAMP largely determine the search performance of Yioop handling queries from a crawl. These are the dictionary subfolder and the posting_doc_shards subfolder, where the former has the greater influence. For the ten million page situation these might be 5GB and 30GB respectively. It is completely possible to copy these subfolders to a SSD and use symlinks to them under the original crawl directory to enhance Yioop's search performance.

Specifying Crawl Options and Modifying Options of the Active Crawl

As we pointed out above, next to the Start Crawl button is an Options link. Clicking on this link, let's you set various aspect of how the next crawl should be conducted. If there is a currently processing crawl, there will be an options link under its stop button. Both of these links lead to similar pages, however, for an active crawl fewer parameters can be changed. So we will only describe the first link. We do mention here though that under the active crawl options page it is possible to inject new seed urls into the crawl as it is progressing. In the case of clicking the Option link next to the start button, the user should be taken to an activity screen which looks like:

The screenshot shows the 'Edit Crawl Options' page in the Yioop Admin interface. The page title is 'Yioop - Admin [Manage Crawls]'. There are two tabs: 'Web Crawl' (selected) and 'Archive Crawl'. The settings are as follows:

- Get Crawl Options From: Use options below
- Server Channel: 0
- Crawl Order: Page Importance
- Max Depth: 2
- Repeat Type: No Repeat
- Crawl Sleep: Start: 10 : 27 AM, Duration: No Sleep
- Robots.txt: Always Follow
- Restrict Sites By Url:
- Disallowed Sites/Sites with Quotas: domainarxiv.org, domainask.com, domainyelp.com, domainclicksense.com
- Seed Sites: https://en.wikipedia.org/wiki/Meghan%2C_Duchess_of_Sussex, http://buzhu.kuaisou.info/robots.txt

A 'Save' button is located at the bottom of the form.

- Blog - Privacy - Terms - ThisSiteBot - Developed at SeekQuarry -
(c) This Site - This Search Engine

The [X] link in the corner returns one to the previous activity. There are two kinds of crawls that can be performed by Yioop either a crawl of sites on the



[Archive.org](#), [100 Crawls](#), [100 Crawls](#), [MediaWiki XWiki Dump](#), [Open Directory Project XML file](#). In the next subsection, we describe new web crawls and then return to archive crawls subsection after that. Finally, we have a short section on some advanced crawl options which can only be set in Config.php or LocalConfig.php. You will probably not need these features but we mention them for completeness.

Web Crawl Options

On the web crawl tab, the first form field, "Get Crawl Options From", allows one to read in crawl options either from the default_crawl.ini file or from the crawl options used in a previous crawl. The rest of the form allows the user to change the existing crawl options. The second form field is labeled **Server Channel**. Each machine in a cluster of Yioop instances with the same Name Server has a channel, defaulting to 0. The Server Channel drop down is populated with a list of channels of currently configured machines in the cluster. If there are no configured machines and empty message is displayed. The Server Channel of a crawl is used to specify which machines in the cluster will participate in the crawl -- only machine with the same channel as that of the crawl will participate. Using this mechanism it is possible to set up several ongoing simultaneous crawls provided they are on different channels.

The **Max Depth** dropdown is used to limit what urls are allowed to be crawl by the number of hops they are from a seed site. For example, if the Max Depth was set to 2, then seed sites would be crawled, sites linked to seed sites would be crawled, and sites linked to sites linked to seed sites would be crawled. A depth 0 crawl only crawls the seed sites.

The next field is labeled **Crawl Order**. This can be set to either Breadth First or Page Importance. It specifies the order in which pages will be crawled. In breadth first crawling, roughly all the seeds sites are visited first, followed by sites linked directly from seed sites, followed by sites linked directly from sites linked directly from seed sites, etc. Page Importance is our modification of [[APC2003](#)]. In this order, each seed sites starts with a certain quantity of money. When a site is crawled it distributes its money equally amongst sites it links to. When picking sites to crawl next, one chooses those that currently have the most money. Additional rules are added to handle things like the fact that some sites might have no outgoing links. Also, in our set-up we don't revisit already seen sites. To handle these situation we take a different tack from the original paper. This crawl order roughly approximates crawling according to page rank.

The **Repeat Type** dropdown controls whether a crawl is a repeating crawl or not, and if so, what is its repeat duration. A non-repeating crawl has one index and crawling continues adding to this index until all allowed urls have been crawled or until the administrator stops the crawl. In a non-repeating crawl one has a double index, that consists of a index to serve search results from and an index to crawl into. Once the repeat time has been exceeded the index that was being crawled into becomes the index to serve results from, the previous search index is reset to empty and is then used to crawl into for the next repeat time amount of time. The **Two Minute** repeat type can be used to experiment with this behavior.

If you are doing an intensive web crawl using your home internet, other family members may from time to time object to not being able to have a good internet connection. The **Crawl Sleep** drop downs allow you to schedule a time during which all the fetcher and queue server processes do not download pages from the internet allow keeping the peace in your household. The first down down is used to specify when daily a crawl sleep period should begin, and the second drop down is used to specify how long it have in hours.

The **Robots Behaviors** dropdown controls the degree to which your Yioop crawler respects **robots.txt** files. A **robots.txt** is a file placed by a site operator in the document root of their web site. I.e., it would typically have a url like: `https://some_host_name/robots.txt`

or

`http://some_host_name/robots.txt`. It is used to specify the files that a particular kind of



- **Always Follow** which always follows to the best of Yioop's abilities the robots.txt instructions.
- **Allow Landing Page Crawl** which allows Yioop to download urls of the form `https://some_host_name/`
or
`http://some_host_name/` but otherwise respects the robots.txt file.
- **Ignore** which allows Yioop to completely ignore the robots.txt file. This option should only be used at your own risk. There might be some use cases such as where you want to crawl part of a site that you yourself own, but where you don't have control of the robots.txt. For the most part, you should not use this option.

The next checkbox is labelled **Restrict Sites by Url** . If it is checked then a textarea with label **Allowed To Crawl Sites** appears. If one checks Restricts Sites by Url then only pages on those sites and domains listed in the Allowed To Crawl Sites textarea can be crawled. We will say how to specify domains and sites in a moment, first let's discuss the last two textareas on the Options form. The Disallowed sites textarea allows you to specify sites that you do not want the crawler to crawl under any circumstance. There are many reasons you might not want a crawler to crawl a site. For instance, some sites might not have a good robots.txt file, but will ban you from interacting with their site if they get too much traffic from you.

Just above the **Seed Sites** textarea are two links "Add User Suggest Data". If on the Server Settings activity Account Registration is set to anything other than Disable Registration, it is possible for a search site user to suggest urls to crawl. This can be done by going to the [Search Tools Page](#) and clicking on the Suggest a Url link. Suggested links are stored in `WORK_DIRECTORY/data/suggest_url.txt`. Clicking Add User Suggest Data adds any suggested urls in this file into the Seed Site textarea, then deletes the contents of this file. The suggested urls which are not already in the seed site list are added after comment lines (lines starting with #) which give the time at which the urls were added. Adding Suggest data can be done either for new crawls or to inject urls into currently running crawls. The Seed sites textarea allows you to specify a list of urls that the crawl should start from. The crawl will begin using these urls. This list can include ".onion" urls if you want to crawl [TOR networks](#).

The format for sites, domains, and urls are the same for each of these textareas, except that the Seed site area can only take urls (or urls and title/descriptions) and in the Allowed To Crawl Sites and Disallowed Sites/Sites with Quotas textareas, one can have domain lines, regex lines, and for Disallowed Sites/Sites with Quotas one can give a url followed by #. In the common format of these textareas, there should be one url, domain, or regex per line with no addition punctuation for separation such as commas. White space is ignored. A domain can be specified as:

```
domain:.sjsu.edu
```

Urls can be specified as:

```
http://www.sjsu.edu/  
https://www.sjsu.edu/gape/  
http://bob.cs.sjsu.edu/index.html
```

would all fall under this domain. The word domain above is a slight misnomer as `domain:sjsu.edu`, without the leading period, also matches a site like `http://mysjsu.edu/`. A site url can be specified as `scheme://domain/path`. Currently, Yioop recognizes the three schemas: `http`, `https`, and `gopher` (an older web protocol). For example, `https://www.somewhere.com/foo/` . Such a site includes `https://www.somewhere.com/foo/anything_more` . Yioop also recognizes `*` and `$` within urls. So `http://my.site.com/*/*` would match `http://my.site.com/subdir1/subdir2/rest` and `http://my.site.com/*/*/$` would require the last symbol in the url to be `'/`. This kind of pattern matching can be useful as another mechanism to restrict the depth of a crawl to within a url to a certain fixed depth --

regex:/some_regular_expression/

For example,

regex:/\./sonico\./

would match any url that contains the string *.sonico*.

In the Disallowed Sites/Sites with Quotas, a number after a # sign indicates that at most that many pages should be downloaded from that site in any given hour. For example,

<http://www.ucanbuyart.com/#100>

indicates that at most 100 pages are to be downloaded from <http://www.ucanbuyart.com/> per hour.

In the seed site area one can specify title and page descriptions for pages that Yioop would otherwise be forbidden to crawl by the robots.txt file. For example,

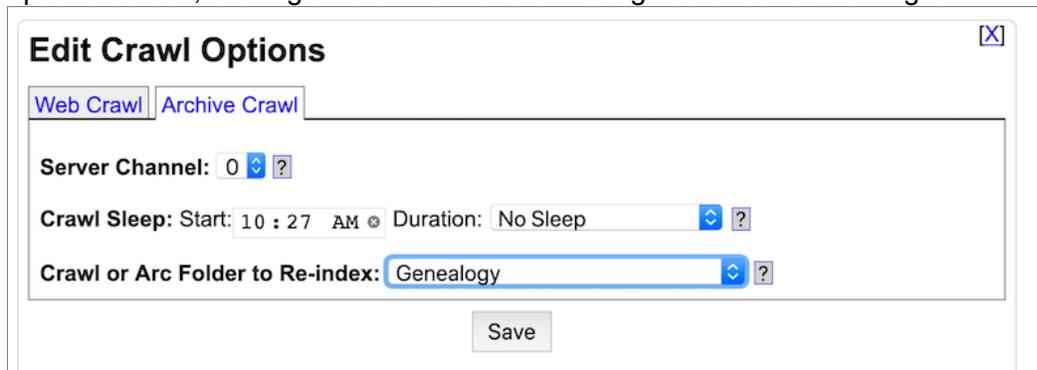
<http://www.facebook.com/###!Facebook###!A%20famous%20social%20media%20site>

tells Yioop to generate a placeholder page for <http://www.facebook.com/> with title "Facebook" and description "A famous social media site" rather than to attempt to download the page. This approach to editing search results is not very flexible and will probably be deprecated. A better approach is to use the [Results Editor](#) activity.

When configuring a new instance of Yioop the file default_crawl.ini is copied to WORK_DIRECTORY/crawl.ini and contains the initial settings for the Options form.

Archive Crawl Options

We now consider how to do crawls of previously obtained archives. From the initial crawl options screen, clicking on the Archive Crawl tab gives one the following form:



Edit Crawl Options [X]

[Web Crawl](#) [Archive Crawl](#)

Server Channel: 0

Crawl Sleep: Start: 10 : 27 AM Duration: No Sleep

Crawl or Arc Folder to Re-index:

The **Server Channel** and **Crawl Sleep** controls are the same as for a web crawl. The **Crawl or Arc folder to Re-Index** dropdown lists all previously done crawls that are available for recrawl.



```
Homepage
test
test
Yahoo
test
meg test
test
test
feed
ARCFILE::English Wikipedia 2012
MIX::images
MIX::videos
MIX::news
MIX::Mix0
MIX::Mix1
MIX::Mix2
MIX::Mix3
```

These include both previously done Yioop crawls, previously down recrawls (prefixed with RECRAWL::), Yioop Crawl Mixes (prefixed with MIX::), and crawls of other file formats such as: arc, warc, database data, MediaWiki XML, and ODP RDF, which have been appropriately prepared in the PROFILE_DIR/cache folder (prefixed with ARCFILE::). In addition, Yioop also has a generic text file archive importer (also, prefixed with ARCFILE::). You might want to re-crawl an existing Yioop crawl if you want to add new meta-words, new cache page links, extract fields in a different manner, or if you are migrating a crawl from an older version of Yioop for which the index isn't readable by your newer version of Yioop. For similar reasons, you might want to recrawl a previously re-crawled crawl. When you archive crawl a crawl mix, Yioop does a search on the keyword site:any using the crawl mix in question. The results are then indexed into a new archive. This new archive might have considerably better query performance (in terms of speed) as compared to queries performed on the original crawl mix. How to make a crawl mix is described in the [Crawl Mixes](#) section. You might want to do an archive crawl of other file formats if you want Yioop to be able to provide search results of their content. Once you have selected the archive you want to crawl, you can add meta words as discussed in the Crawl Time Tab Page Rule portion of the [Page Options](#) section. Afterwards, go back to the Create Crawl screen to start your crawl. As with a Web Crawl, for an archive crawl you need both the queue server running and a least one fetcher running to perform a crawl.

To re-crawl a previously created web archive or mix that was made using several fetchers, each of the fetchers that was used in the creation process should be running. In addition, **a recrawl is only possible if the original crawl (or crawls in the mix case) were crawled with the Cache whole crawled pages option set to true**. The same fetchers restriction is because the data used in the recrawl will come locally from the machine of that fetcher. For other kinds of archive crawls, which fetchers one uses, doesn't matter because archive crawl data comes through the name server. You might also notice that the number of pages in a web archive re-crawl is actually larger than the initial crawl. This can happen because during the initial crawl data was stored in the fetcher's archive bundle and a partial index of this data sent to appropriate queue servers but was not yet processed by these queue servers. So it was waiting in a schedules folder to be processed in the event the crawl was resumed.

To get Yioop to detect arc, database data, MediaWiki, ODP RDF, or generic text archive files you need to create a WORK_DIRECTORY/archives folder (if it doesn't already exists) on the name server machine. Yioop checks subfolders of this for files with the name arc_description.ini. For example, to do a Wikimedia archive crawl, one could make a subfolder WORK_DIRECTORY/archives/my_wiki_media_files and put in it a file arc_description.ini in the format to be discussed in a moment. In addition to the arc_description.ini, you would also put in this folder all the archive files (or links to them)



Specify the kind of archives the folder contains and how to extract it. An example arc_description.ini for MediaWiki dumps might look like:

```
arc_type = 'MediaWikiArchiveBundle';
description = 'English Wikipedia 2012';
```

In the Archive Crawl dropdown the description will appear with the prefix ARCFILE:: and you can then select it as the source to crawl. Currently, the supported arc_types are: ArcArchiveBundle, DatabaseBundle, MediaWikiArchiveBundle, OdpRdfArchiveBundle, TextArchiveBundle, and WarcArchiveBundle. For the ArcArchiveBundle, OdpRdfArchiveBundle, MediaWikiArchiveBundle, WarcArchiveBundle arc_types, generally a two line arc_description.ini file like above suffices. We now describe how to import from the other kind of formats in a little more detail. In general, the arc_description.ini will tell Yioop how to get string items (in a associative array with a minimal amount of additional information) from the archive in question. Processing on these string items can then be controlled using Page Rules, described in the [Page Options](#) section.

An example arc_description.ini where the arc_type is DatabaseBundle might be:

```
arc_type = 'DatabaseBundle';
description = 'DB Records';
dbms = "mysql";
db_host = "localhost";
db_name = "MYGREATDB";
db_user = "someone";
db_password = "secret";
encoding = "UTF-8";
sql = "SELECT MYCOL1, MYCOL2 FROM MYTABLE1 M1, MYTABLE2 M2 WHERE M1.FOO=M2.BAR";
field_value_separator = '|';
column_separator = '##';
```

Here is a specific example that gets the rows out of the TRANSLATION table of Yioop where the database was stored in a Postgres DBMS. In the comments I indicate how to alter it for other DBMS's.

```
arc_type = 'DatabaseBundle';
description = 'DB Records';
;sqlite3 specific
;dbms ="sqlite3";
;mysql specific
;dbms = "mysql";
;db_host = "localhost";
;db_user = "root";
;db_password = "";
dbms = "pdo";
;below is for postgres; similar if want db2 or oracle
db_host = "pgsql:host=localhost;port=5432;dbname=seek_quarry"
db_name = "seek_quarry";
db_user = "cpollett";
db_password = "";
encoding = "UTF-8";
sql = "SELECT * from TRANSLATION";
field_value_separator = '|';
column_separator = '##';
```

Possible values for dbms are pdo, mysql, sqlite3. If pdo is chosen, then db_host should be a PHP DSN specifying which DBMS driver to use. db_name is the name of the database you would like to connect to, db_user is the database username, db_password is the password for that user, and encoding is the character set of rows that the database query will return.

The sql variable is used to give a query whose result rows will be the items indexed by Yioop. Yioop indexes string "pages", so to make these rows into a string each column result will be made into a string: field field_value_separator value. Here field is the name of the column, value is the value for that column in the given result row. Columns are concatenated together separated by the value of of column_separator. The resulting string is then sent to Yioop's TextProcessor page processor.

We next give a few examples of arc_description.ini files where the arc_type is TextArchiveBundle. First, suppose we wanted to index access log file records that look like:

```
127.0.0.1 - - [21/Dec/2012:09:03:01 -0800] "POST /git/yioop2/ HTTP/1.1" 200 - \
```



then might use the following arc_description.ini file:

```
arc_type = 'TextArchiveBundle';  
description = 'Log Files';  
compression = 'plain';  
file_extension = 'log';  
end_delimiter = "\n";  
encoding = "UTF-8";
```

In addition to compression = 'plain', Yioop supports gzip and bzip2. The end_delimiter is a regular expression indicating how to know when a record ends. To process a TextArchiveBundle Yioop needs either an end_delimiter or a start_delimiter (or both) to be specified. As another example, for a mail.log file with entries of the form:

```
From pollett@mathcs.sjsu.edu Wed Aug 7 10:59:04 2002 -0700  
Date: Wed, 7 Aug 2002 10:59:04 -0700 (PDT)  
From: Chris Pollett <pollett@mathcs.sjsu.edu>  
X-Sender: pollett@eniatic.cs.sjsu.edu  
To: John Doe <johndoe@mail.com>  
Subject: Re: a message  
In-Reply-To: <5.1.0.14.0.20020723093456.00ac9c00@mail.com>  
Message-ID: <Pine.G50.4.05.10208071057420.9463-100000@eniatic.cs.sjsu.edu>  
MIME-Version: 1.0  
Content-Type: TEXT/PLAIN; charset=US-ASCII  
Status: 0  
X-Status:  
X-Keywords:  
X-UID: 17
```

Hi John,

I got your mail.

Chris

The following might be used:

```
arc_type = 'TextArchiveBundle';  
description = 'Mail Logs';  
compression = 'plain';  
file_extension = 'log';  
start_delimiter = "\n\nFrom\s";  
encoding = "ASCII";
```

Notice here we are splitting records using a start delimiter. Also, we have chosen ASCII as the character encoding. As a final example, we show how to import tar gzip files of Usenet records as found, in the [UTzoo Usenet Archive 1981-1991](#). Further discussion on how to process this collection is given in the [Page Options](#) section.

```
arc_type = 'TextArchiveBundle';  
description = 'Utzoo Usenet Archive';  
compression = 'gzip';  
file_extension = 'tgz';  
start_delimiter = "\\0\\0\\0\\0Path:";  
end_delimiter = "\\n\\0\\0\\0";  
encoding = "ASCII";
```

Notice in the above we set the compression to be gzip. Then we have Yioop act on the raw tar file. In tar files, content objects are separated by long paddings of null's. Usenet posts begin with Path, so to keep things simple we grab records which begin with a sequence of null's the Path and end with another sequence of null's.

As a final reminder for this section, remember that, in addition, to the arc_description.ini file, the subfolder should also contain instances of the files in question that you would like to archive crawl. So for arc files, these would be files of extension .arc.gz; for MediaWiki, files of extension .xml.bz2; and for ODP-RDF, files of extension .rdf.u8.gz. Crawl Options of Config.php or LocalConfig.php

There are a couple of flags which can be set in the Config.php or in a LocalConfig.php file that affect web crawling which we now mention for completeness. As was mentioned before, when Yioop is crawling it makes use of Etag: and Expires: HTTP headers received during web page download to determine when a page can be recrawled. This assumes one



```
nsdefine("USE_ETAG_EXPIRES", false);
```

[Return to table of contents](#)

Mixing Crawl Indexes

Once you have performed a few crawls with Yioop, you can use the Mix Crawls activity to create mixture of your crawls. This section describes how to create crawl mixes which are processed when a query comes in to Yioop. Once one has created such a crawl mix, an admin user can make a new index which consists of results of the crawl mix ("materialize it") by doing an archive crawl of the crawl mix. The [Archive Crawl Options](#) subsection has more details on how to do this latter operation. The main Mix Crawls activity looks like:

The screenshot shows the 'Admin [Mix Crawls]' page. At the top, there's a header with the Yioop logo and the title. Below that, a table titled 'Mixes' is displayed. The table has columns for 'Name', 'Definition', and 'Actions'. There are two rows: 'Mix0' and 'Mix1'. 'Mix0' has a definition of '#1{1.0 * (Genealogy[ASC] + K:)}' and 'Mix1' has '#1{1.0 * (IBM[ASC] + K:)}'. The 'Actions' column for each row contains 'Edit', 'Set as Index', and 'Delete' links. Above the table, there's a pagination control showing 'Row 0 to 50 of 103' and a 'Show 50' dropdown.

The add icon on this activity take you to a form that allows you to name and create a new, empty crawl mixture. This mix would then appear in the Mixes table that one can see in the image above. Each row in the Mixes table corresponds to one crawl mix. The first column has the name of the mix, the second column says how the mix is built out of component crawls, and the actions columns allows you to edit the mix, set it as the default index for Yioop search results, or delete the mix. You can also append "m:name+of+mix" or "mix:name+of+mix" to a query to use that crawl mix without having to set it as the index. Editing an existing mix sends you to a second page:

The screenshot shows the 'Edit Crawl Mix' form. At the top, there's a header with the Yioop logo and the title. Below that, there's a form with a 'Mix Name' field containing 'Forward Facing Mix'. Underneath, there's a section for 'Mix Components' with a link to 'Add Search Result Fragment'. There are two tables of components. The first table has columns for 'Weight', 'Name', 'Order', 'Keywords', and 'Actions'. It contains one row with '1' weight, 'feed' name, 'Descending' order, and 'media:news' keywords. The second table is similar but contains two rows: '1' weight, 'Default Crawl' name, 'Ascending' order, and 'Homepage' name. At the bottom of the form, there's a 'Save' button.

Using the [X] link on this page will take you to the prior screen. The first text field on the edit page lets you rename your mix if you so desire. Beneath this is an "Add Search Result Fragment" link. A fragment is a weighted list of crawls. When search results are calculated



way to treat position as a score is ascending, earlier items are more important. For some index sources such as the news feed seen in the example above later, more recent items should be more important, so position should be treated as a score in a descending fashion. This is what the order column dropdowns control. A given item may also appear in more than one crawl listed in a fragment. The weighting then says how much to weigh that particular crawl's score when computing a total score for an item. For example, in the image above scores from the homepage crawl would be doubled and added to score from the default crawl to give total score. Results in a fragment would then be output in the order of decreasing total score. If only one fragment were present, then search results would come from any crawl listed for this fragment. If more than one group is present then the number of results field for that group determines how many of the displayed results should come from that group. For the Crawl Mix displayed above, there are two groups: The first group is used to display the first three results, the second group indicates that a 100 results should come from it, however, as it is the last fragment listed all remaining results will come from it. For some indexes, we may want to take the input query and add query terms to it before querying that index. This is done using the keywords column of the Edit Mix. As we can see in the above, for the fragment with the feed index we add the meta keyword `media:news` to whatever the base query was to further restrict items that come from that index. Keywords can be used to make a particular component of a crawl mix behave in a conditional many by using the "if:" meta word described in the search and user interface section. The last link in a crawl row allows you to delete a crawl from a crawl group. For changes on this page to take effect, the "Save" button beneath this dropdown must be clicked.

[Return to table of contents](#) .

Classifying Web Pages

Sometimes searching for text that occurs within a page isn't enough to find what one is looking for. For example, the relevant set of documents may have many terms in common, with only a small subset showing up on any particular page, so that one would have to search for many disjoint terms in order to find all relevant pages. Or one may not know which terms are relevant, making it hard to formulate an appropriate query. Or the relevant documents may share many key terms with irrelevant documents, making it difficult to formulate a query that fetches one but not the other. Under these circumstances (among others), it would be useful to have meta words already associated with the relevant documents, so that one could just search for the meta word. The Classifiers activity provides a way to train classifiers that recognize classes of documents; these classifiers can then be used during a crawl to add appropriate meta words to pages determined to belong to one or more classes.

Clicking on the Classifiers activity displays a text field where you can create a new classifier, and a table of existing classifiers, where each row corresponds to a classifier and provides some statistics and action links. A classifier is identified by its class label, which is also used to form the meta word that will be attached to documents. Each classifier can only be trained to recognize instances of a single target class, so the class label should be a short description of that class, containing only alphanumeric characters and underscores (e.g., "spam", "homepage", or "menu"). Typing a new class label into the text box and hitting the Create button initializes a new classifier, which will then show up in the table.

[root]

 [Admin](#) [Manage Classifiers]

Classifiers  Row 0 to 2 of 2 Show 50  

Label	Positive	Negative	Actions		
humanintereststory <small>21 Aug 2020 09:47:42</small>	581	587	Edit	Finalized	Delete
spam <small>21 Aug 2020 09:19:50</small>	988	971	Edit	Finalized	Delete



action link is used to prepare a classifier to classify new web pages, which cannot be done until you've added some training examples. We'll discuss how to add new examples next, then return to the Finalize link.

Editing a Classifier

Clicking on the Edit action link takes you to a new page where you can change a classifier's class label, view some statistics, and provide examples of positive and negative instances of the target class. The first two options should be self-explanatory, but the last is somewhat involved. A classifier needs labeled training examples in order to learn to recognize instances of a particular class, and you help provide these by picking out example pages from previous crawls and telling the classification system whether they belong to the class or do not belong to the class. The Add Examples section of the Edit Classifier page lets you select an existing crawl to draw potential examples from, and optionally narrow down the examples to those that satisfy a query. Next to the label Source in the screenshot one can see two dropdowns: one let's you choose a previous crawl to get examples from, the other let's you either add all example from this crawl as in the class, none from this crawl as in the class, or specify that you want to hand label pages from this crawl. Once you've selected your choices for these drop downs, clicking the Load button will send a request to the server to load some pages from the crawl. If you chose hand labeling, a next search result to label will be presented. This will come with several action links along the side that let you mark this document as either a positive or negative example of the target class, or skip this document and move on to the next one:

Edit Classifier [X]

Classifier Label:

Statistics

Positive Examples: 3
Negative Examples: 1
Accuracy: N/A

Add Examples

Source:

Keywords:

No documents

<p><input type="button" value="[In Class]"/></p> <p><input type="button" value="[Not In Class]"/></p> <p><input type="button" value="[Skip]"/></p>	<p>Yahoo</p> <p>https://www.yahoo.com/</p> <p>Prediction: homepage (60.0% confidence, 65.9% disagreement)</p> <p>.. News, email and search are just the beginning. Discover more every day. Find your yodel. Yahoo . Search . Skip to Navigation Skip to Main Content Skip to Related Content Sign in . Mail Sign in to view your mail. Mail () Coronavirus News Finance Sports Politics Black Lives Matter Entertainment Life StayHome Subscriptions More...</p>
<p><input type="button" value="[In Class]"/></p> <p><input type="button" value="[Not In Class]"/></p> <p><input type="button" value="[Skip]"/></p>	<p>Chris Pollett > Students > Qiao Liu</p> <p>http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Spring17/qiao/index.php?Bio.php</p> <p>Prediction: homepage (50.0% confidence, 70.8% disagreement)</p> <p>Chris Pollett > Students > Qiao Liu Chris</p>
<p><input type="button" value="[In Class]"/></p> <p><input type="button" value="[Not In Class]"/></p> <p><input type="button" value="[Skip]"/></p>	<p>Chris Pollett > Students > Youn Kim</p> <p>http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Spring10/youn/index.php?Bio.php</p> <p>Prediction: homepage (33.3% confidence, 74.7% disagreement)</p> <p>Chris Pollett > Students > Youn Kim . Chris</p>
<p><input type="button" value="[In Class]"/></p> <p><input type="button" value="[Not In Class]"/></p>	<p>Chris Pollett's Homepage</p> <p>https://www.cs.sjsu.edu/faculty/pollett/</p>

When you select any of the action buttons, your choice is sent back to the server, and a new example to label is sent back (so long as there are more examples in the selected index). The old example record is shifted down the page and its background color updated to reflect your decision—green for a positive example, red for a negative one, and gray for a skip; the statistics at the top of the page are updated accordingly. The new example record replaces the old one, and the process repeats. Each time a new label is sent to the server, it is added to the training set that will ultimately be used to prepare the classifier to classify new web pages during a crawl. Each time you label a set number of new examples



labeled). The new estimated accuracy, calculated as the proportion of the test pages classified correctly, is displayed under the Statistics section. You can also manually request an updated accuracy estimate by clicking the Update action link next to the Accuracy field. Doing this will send a request to the server that will initiate the same process described previously, and after a delay, display the new estimate.

All of this happens without reloading the page, so avoid using the web browser's Back button. If you do end up reloading the page somehow, then the current example record and the list of previously-labeled examples will be gone, but none of your progress toward building the training set will be lost.

Finalizing a Classifier

Editing a classifier adds new labeled examples to the training set, providing the classifier with a more complete picture of the kinds of documents it can expect to see in the future. In order to take advantage of an expanded training set, though, you need to finalize the classifier. This is broken out into a separate step because it involves optimizing a function over the entire training set (for web-based, created classifiers it does lasso regression on the 30 most χ^2 -informative terms in the summaries of the pages of the training set), which can be slow for even a few hundred example documents. It wouldn't be practical to wait for the classifier to re-train each time you add a new example, so you have to explicitly tell the classifier that you're done adding examples for now by clicking on the Finalize action link either next to the Load button on the edit classifier page or next to the given classifier's name on the classifier management page.

Clicking this link will kick off a separate process that trains the classifier in the background. When the page reloads, the Finalize link should have changed to text that reads

"Finalizing..." (but if the training set is very small, training may complete almost immediately). After starting finalization, it's fine to walk away for a bit, reload the page, or carry out some unrelated task for the user account. You should not however, make further changes to the classifier's training set, or start a new crawl that makes use of the classifier. When the classifier finishes its training phase, the Finalizing message will be replaced by one that reads "Finalized" indicating that the classifier is ready for use.

As a general tip in creating your own classifiers, it is easier to use the *all in class/none in class* settings rather than hand label examples. One easier way to make a classifier is to prepare a list of urls you know are in the class you want and do a depth 1 crawl of these pages. Then prepare a list of urls you know are not in the class you want and do a depth 1 crawl of these pages. Finally, use these two crawls to make your classifier using the GUI just described.

Using a Classifier

Using a classifier is as simple as checking the "Use to Classify" or "Use to Rank" checkboxes next to the classifier's label on the [Page Options](#) activity, under the "Classifiers and Rankers" heading. When the next crawl starts, the classifier (and any other selected classifiers) will be applied to each fetched page. If "Use to Rank" is checked then the classifier score for that page will be recorded. If "Use to Classify" is checked and if a page is determined to belong to a target class, it will have several meta words added. As an example, if the target class is "spam", and a page is determined to belong to the class with probability .79, then the page will have the following meta words added:

- class:spam
- class:spam:50plus
- class:spam:60plus
- class:spam:70plus
- class:spam:70

These meta words allow one to search for all pages classified as spam at any probability over the preset threshold of .50 (with class:spam), at any probability over a specific multiple of .1 (e.g., over .6 with class:spam:60plus), or within a specific range (e.g., .60–.69 with

[Return to table of contents](#).

Page Indexing and Search Options

Several properties about how web pages are indexed and how pages are looked up at search time can be controlled by clicking on Page Options. There are three tabs for this activity: Crawl Time, Search Time, and Test Options. We will discuss each of these in turn.

Crawl Time Tab

Clicking on Page Options leads to the Crawl Time Tab:



Get Page Options Form. Use options below

Byte Range to Download (0 - Value): ?

Summarizer: ?

Max Page Summary Length in Bytes: ?

Cache Whole Crawled Pages:

Allow Page Recrawl After: ?

Page File Types to Crawl:

unknown <input checked="" type="checkbox"/>	asp <input checked="" type="checkbox"/>	java <input checked="" type="checkbox"/>
bmp <input checked="" type="checkbox"/>	aspx <input checked="" type="checkbox"/>	jpg <input checked="" type="checkbox"/>
gz <input checked="" type="checkbox"/>	cgi <input checked="" type="checkbox"/>	jpeg <input checked="" type="checkbox"/>
bz <input checked="" type="checkbox"/>	cfm <input checked="" type="checkbox"/>	pdf <input checked="" type="checkbox"/>
csv <input checked="" type="checkbox"/>	cfml <input checked="" type="checkbox"/>	png <input checked="" type="checkbox"/>
tab <input checked="" type="checkbox"/>	do <input checked="" type="checkbox"/>	ppt <input checked="" type="checkbox"/>
tsv <input checked="" type="checkbox"/>	htm <input checked="" type="checkbox"/>	pptx <input checked="" type="checkbox"/>
txt <input checked="" type="checkbox"/>	html <input checked="" type="checkbox"/>	rss <input checked="" type="checkbox"/>
doc <input checked="" type="checkbox"/>	jsp <input checked="" type="checkbox"/>	rtf <input checked="" type="checkbox"/>
docx <input checked="" type="checkbox"/>	php <input checked="" type="checkbox"/>	svg <input checked="" type="checkbox"/>
epub <input checked="" type="checkbox"/>	pl <input checked="" type="checkbox"/>	xlsx <input checked="" type="checkbox"/>
gif <input checked="" type="checkbox"/>	py <input checked="" type="checkbox"/>	
xml <input checked="" type="checkbox"/>	shtml <input checked="" type="checkbox"/>	

Classifiers and Rankers ?

Row 0 to 3 of 3 Show ?

	Use to Classify	Use to Rank
homepage	<input type="checkbox"/>	<input type="checkbox"/>
humanintereststory	<input type="checkbox"/>	<input type="checkbox"/>
spam	<input type="checkbox"/>	<input type="checkbox"/>

Indexing Plugins ?

Plugin	Use in Crawl
AddressesPlugin	<input type="checkbox"/>
RecipePlugin	<input type="checkbox"/>
WordfilterPlugin	<input type="checkbox"/> [Configure]

Page Field Extraction Rules ?

Save

This tab controls some aspects about how a page is processed and indexed at crawl time. The form elements before Page Field Extraction Rules are relatively straightforward and we will discuss these briefly below. The Page Rules textarea allows you to specify additional commands for how you would like text to be extracted from a page document summary. The description of this language will take the remainder of this subsection.



requirements on disk space needed for a crawl, bigger numbers would tend to improve the search results. If whole pages are being cached, these downloaded bytes are stored in archives with the fetcher. The Summarizer dropdown control what summarizer is used on a page during page processing. Yioop uses a summarizer to control what portions of a page will be put into the index and are available at search time for snippets. The four available summarizers are Basic, Centroid, Centroid Weighted, and Graph Based. A description for what each of these does can be found by clicking on the Help Wiki [?] next to this drop down. At high level, Basic uses tag scraping together with distance from the top of the document to make its summary, the two Centroid methods, split the document into sentences, calculate different kinds of average sentences, and use these to determine what sentences should be added to the summary, finally, Graph-Based, uses a page rank style approach where a weighted adjacency matrix between sentences is calculated using a notion of similarity between two sentences. How to tweak the Centroid summarizers for a particular locale, is described in the [Localizing Yioop](#) section. The Max Page Summary Length in Bytes controls how many of the total bytes can be used to make a page summary which is sent to the queue server. It is only words in this summary which can actually be looked up in search result. Care should be taken in making this value larger as it can increase the both the RAM memory requirements (you might have to change the `memory_limit` variable at the start of `QueueServer.php` to prevent crashing) while crawling and it can slow the crawl process down. The Cache whole crawled pages checkbox says whether to when crawling to keep both the whole web page downloaded as well as the summary extracted from the web page (checked) or just to keep the page summary (unchecked). The Suffix Phrases checkbox controls whether suffix phrases are extracted from document summaries at crawl time and whether suffix phrases are used at search time. Using suffix phrases allows for faster, by building a complete suffix tree of the corpus, but often less relevant, search results for multi-term queries in the spin hard drive setting for a very large index (> 100 million pages). The resulting index is substantially larger if Suffix Phrases are used, so by default this checkbox is off. The next dropdown, Allow Page Recrawl After, controls how many days that Yioop keeps track of all the URLs that it has downloaded from. For instance, if one sets this dropdown to 7, then after seven days Yioop will clear its Bloom Filter files used to store which urls have been downloaded, and it would be allowed to recrawl these urls again if they happened in links. It should be noted that all of the information from before the seven days will still be in the index, just that now Yioop will be able to recrawl pages that it had previously crawled. Besides letting Yioop get a fresher version of page it already has, this also has the benefit of speeding up longer crawls as Yioop doesn't need to check as many Bloom filter files. In particular, it might just use one and keep it in memory.

The Page File Types to Crawl checkboxes allow you to decide which file extensions you want Yioop to download during a crawl. This check is done before any download is attempted, so Yioop at that point can only guess the [MIME Type](#), as it hasn't received this information from the server yet. An example of a url with a file extension is:

```
http://flickr.com/humans.txt
```

which has the extension txt. So if txt is unchecked, then Yioop won't try to download this page even though Yioop can process plain text files. A url like:

```
http://flickr.com/
```

has no file extension and will be assumed to be have a html extension. To crawl sites which have a file extension, but no one in the above list check the unknown checkbox in the upper left of this list.

The Classifiers and Rankers checkboxes allow you to select the classifiers that will be used to classify or rank pages. Each classifier (see the [Classifiers](#) section for details) is represented in the list by its class label and two checkboxes. Checking the box under Use to classify indicates that the associated classifier should be used (made active) during the next crawl for classifying, checking the "Use to Rank" indicates that the classifier should be



...belong to the class that the classifier has been trained to recognize, then a meta word like "class:label", where label is the class label, is added to the page summary. For faster access to pages that contain a single term and a label, for example, pages that contain "rich" and are labeled as "non-spam", Yioop actually uses the first character of the label "non-spam" and embeds it as part of the term ID of "rich" on "non-spam" pages with the word "rich". To ensure this speed-up can be used it is useful to make sure ones classifier labels begin with different first characters. If "Use to Rank" is checked then when a classifier is run on the page, the score from the classifier is recorded. When a search is done that might retrieve this page, this score is then used as one component of the overall score that this page receives for the query.

The Indexing Plugins checkboxes allow you to select which plugins to use during the crawl. Yioop comes with three built-in plugins: AddressesPlugin, RecipePlugin, and WordFilterPlugin. One can also write or download additional plugins. If the plugin can be configured, next to the checkbox will be a link to a configuration screen. Let's briefly look at each of these plugins in turn...

Checking the AddressesPlugin enables Yioop during a crawl to try to calculate addresses for each page summary it creates. When Yioop processes a page it by default creates a summary of the page with a TITLE and a DESCRIPTION as well as a few other fields. With the addresses plugin activated, it will try to extract data to three additional fields: EMAILS, PHONE_NUMBERS, and ADDRESSES. If you want to test out how these behave, pick some web page, view source on the web page, copy the source, and then paste into the Test Options Tab on the page options page (the Test Options Tab is described later in this section). The address plugin also adds two meta words email: and phone: which can be used to search for pages with a particular email or phone number. The address plugin is further discussed in the *Page Field Extraction Language* subsection below where an example is given of using it to generate a file of street addresses found during a crawl.

Clicking the RecipePlugin checkbox causes Yioop during a crawl to run the code in indexing_plugins/recipe_plugin.php. This code tries to detect pages which are food recipes and separately extracts these recipes and clusters them by ingredient. It then add search meta words ingredient: and recipe:all to allow one to search recipes by ingredient or only documents containing recipes.

Checking the WordFilterPlugin causes Yioop to run code in indexing_plugins/wordfilter_plugin.php on each downloaded page. The [Niche Crawling Video Tutorial](#) has information about how to use this plugin to create subject-specific crawls of the web. This code checks if the downloaded page has one of the words listed in the textarea one finds on the plugin's configure page. If it does, then the plugin follows the actions listed for pages that contain that term. Below is an example WordFilterPlugin configure page:

```
;/ Below is a set of dummy word filter rules to be used with the zip file
;/ tests/word-filter-test-crawl.zip . A description of how to carry out this
;/ test is given there in the readme.txt file in this zip file.
;/ You should not use these rules on an actual web crawl or you will get very
;/ minimal results.
;/
;/ The general format of this word filter rule set is a sequence of rule blocks:
;/ [some_url_or_domain1]
;/ rule_1
;/ rule_2
;/ ...
;/ [some_url_or_domain2]
;/
;/ Anything on a line after a ';' is also treated as a comment.
;/ A rule block begins with a url or domain that the rules that follow it are
;/ to apply to. urls and domains follow the site format as used in crawl
;/ options. For example, http://somesite.com/sub_dir/file or domain:somewhere.com.
;/ A minus in front of this url can be used to indicate anything but that domain.
;/ For example, -domain:ca would mean anything but the ca domain.
;/ The rules given before any [some_url_or_domain] line are applied to ALL
;/ summaries. When a summaries is being processed, the set of applicable rule
;/ blocks is determined based on matching the summaries url with
;/ the rule block urls, then all rules from these blocks are applied. Here is
;/ an example set of rules which is roughly suitable to return only pages with
;/ Canadian content on Wikipedia:
;/ [domain:wikipedia.org]
;/ -canada,-canadian:NOPROCESS
```

Lines in this configure file either specify a url or domain using a syntax like [url_or_domain] or specify a rule or a comment. Whitespace is ignored and everything after a semi-colon on a line is treated as a comment. The rules immediately following a url or domain line up till the next url or domain line are in effect if one crawling is crawling a page with that url or domain. Each **rule line** in the textarea consists of a comma separated list of literals followed by a colon followed by a comma separated list of what to do if the literal condition is satisfied. A single literal in the list of literals is an optional + or - followed by a sequence of non-space characters. After the + or -, up until a ; symbol is called the term in the literal. If the literal sign is + or if no sign is present, then the literal holds for a document if it contains the term, if the literal sign is - then the literal holds for a document if it does not contain the term, if there is a decimal number between 0 and 1, say x, after the # up to a comma or the first white-space character, then this is modified so the literal holds only if x'th fraction of the documents length comes from the literal's term. If rather than a decimal x were a positive natural number then the term would need to occur x times. If all the literal in the comma separated list hold, then the rule is said to hold, and the actions will apply. The line -term0:JUSTFOLLOW says that if the downloaded page does not contain the word "term0" then do not index the page, but do follow outgoing links from the page. The line term1:NOPROCESS says if the document has the word "term1" then do not index it or follow links from it. The last line +term2:NOFOLLOW,NOSNIPPET says if the page contains "term2" then do not follow any outgoing links. NOSNIPPET means that if the page is returned from search results, the link to the page should not have a snippet of text from that page beneath it. As an example of a more complicated rule, consider:

```
surfboard#2,bikini#0.02:NOINDEX, NOFOLLOW
```

Here for the rule to hold the condition surfboard#2 requires that the term surfboard occurred at least twice in the document and the condition bikini#0.02 requires that 0.02 percent of the documents total length also come from copies of the word bikini. In addition, to the commands just mentioned, WordFilterPlugin supports standard robots.txt directives such as: NOINDEX, NOCACHE, NOARCHIVE, NOODP, NOYDIR, and NONE. More details about how indexing plugins work and how to write your own indexing plugin can be found in the [Modifying Yioop](#) section.

Page Field Extraction Language

We now return to the Page Field Extraction Rules textarea of the Page Options - Crawl Time tab. Commands in this area allow a user to control what data is extracted from a



useful for extracting data from generic text archives and database archives. How to import such archives is described in the Archive Crawls sub-section of [Performing and Managing Crawls](#). The input to the page rule processor is an associative array that results from Yioop doing initial processing on a page. To see what this array looks like one can take a web page and paste it into the form on the Test Options tab. There are two types of page rule statements that a user can define: command statements and assignment statements. In addition, a semicolon ';' can be used to indicate the rest of a line is a comment. Although the initial textarea for rules might appear small. Most modern browsers allow one to resize this area by dragging on the lower right hand corner of the area. This makes it relatively easy to see large sets of rules.

A command statement takes a key field argument for the page associative array and does a function call to manipulate that page. Below is a list of currently supported commands followed by comments on what they do:

```

addMetaWords(field)      ;add the field and field value to the META_WORD
                          ;array for the page
addKeywordLink(field)    ;split the field on a comma, view this as a search
                          ;keywords => link text association, and add this to
                          ;the KEYWORD_LINKS array.
setStack(field)          ;set which field value should be used as a stack
pushStack(field)         ;add the field value for field to the top of stack
popStack(field)          ;pop the top of the stack into the field value for
                          ;field
setOutputFolder(dir)     ;if auxiliary output, rather than just to the
                          ;a yioop index, is being done, then set the folder
                          ;for this output to be dir
setOutputFormat(format) ;set the format of auxiliary output.
                          ;Should be either CSV or SQL
                          ;SQL mean that writeOutput will write an insert
                          ;statement
setOutputTable(table)   ;if output is SQL then what table to use for the
                          ;insert statements
toArray(field)           ;splits field value for field on a comma and
                          ;assign field value to be the resulting array
toString(field)         ;if field value is an array then implode that
                          ;array using comma and store the result in field
                          ;value
unset(field)            ;unset that field value
writeOutput(field)      ;use the contents of field value viewed as an array
                          ;to fill in the columns of a SQL insert statement
                          ;or CSV row

```

Page rule assignments can either be straight assignments with '=' or concatenation assignments with '.='. Let \$page indicate the associative array that Yioop supplies the page rule processor. There are four kinds of values that one can assign:

```

field = some_other_field ; sets $page['field'] = $page['some_other_field']
field = "some_string" ; sets $page['field'] to "some string"
field = /some_regex/replacement_where_dollar_vars_allowed/
; computes the results of replacing matches to some_regex
; in $page['field'] with replacement_where_dollar_vars_allowed
field = /some_regex/g ;sets $page['field'] to the array of all matches
; of some regex in $page['field']

```

For each of the above assignments we could have used ".=" instead of "=". We next give a simple example and followed by a couple more complicated examples of page rules and the context in which they were used:

In the first example, we just want to extract meaningful titles for mail log records that were read in using a TextArchiveBundleIterator. Here after initial page processing a whole email would end up in the DESCRIPTION field of the \$page associative array given tot the page rule processor. So we use the following two rules:

```

TITLE = DESCRIPTION
TITLE = /(.\n|\Z)*?Subject:[\t ](.+?)\n(.\n|\Z)*?$/

```

We initially set the TITLE to be the whole record, then use a regex to extract out the correct portion of the subject line. Between the first two slashes recognizes the whole record where the pattern inside the second pair of parentheses (.+?) matches the subject text. The \$2 after the second parenthesis says replace the value of TITLE with just this portion.



```
;
; Set the UI_FLAGS variable. This variable in a summary controls
; which of the header elements should appear on cache pages.
; UI_FLAGS should be set to a string with a comma separated list
; of the options one wants. In this case, we use: yioop_nav, says that
; we do want to display header; version, says that we want to display
; when a cache item was crawled by Yioop; and summaries, says to display
; the toggle extracted summaries link and associated summary data.
; Other possible UI_FLAGS are history, whether to display the history
; dropdown to other cached versions of item; highlight, whether search
; keywords should be highlighted in cached items
;
UI_FLAGS = "yioop_nav,version,summaries"
;
; Use Post Subject line for title
;
TITLE = DESCRIPTION
TITLE = /(.\|\n)*?Subject:([\^\n]+\)\n(.\|\n)*$/2/
;
; Add a link with a blank keyword search so cache pages have
; link back to yioop
;
link_yioop = ",Yioop"
addKeywordLink(link_yioop)
unset(link_yioop) ;using unset so don't have link_yioop in final summary
;
; Extract y-M and y-M-j dates as meta word u:date:y-M and u:date:y-M-j
;
date = DESCRIPTION
date = /(.\|\n)*?Date:([\^\n]+\)\n(.\|\n)*$/2/
date = /.*,\s*(\d*)-(\w*)-(\d*)\s*.*$/3-2-1/
addMetaWord(date)
date = /(\d*)-(\w*)-.*$/1-2/
addMetaWord(date)
;
; Add a link to articles containing u:date:y-M meta word. The link text
; is Date:y-M
;
link_date = "u:date:"
link_date .= date
link_date .= ",Date:"
link_date .= date
addKeywordLink(link_date)
;
; Add u:date:y meta-word
;
date = /(\d*)-.*$/1/
addMetaWord(date)
;
; Get the first three words of subject ignoring re: separated by underscores
;
subject = TITLE
subject = /(\s*(RE|re|RE|Re)\s*)?(.*)/$3/
subject_word1 = subject
subject_word1 = /\s*([\^\s]*).*/$1/
subject_word2 = subject
subject_word2 = /\s*([\^\s]*)\s*([\^\s]*).*/$2/
subject_word3 = subject
subject_word3 = /\s*([\^\s]*)\s*([\^\s]*)\s*([\^\s]*).*/$3/
subject = subject_word1
unset(subject_word1)
subject .= "_ "
subject .= subject_word2
unset(subject_word2)
subject .= "_ "
subject .= subject_word3
unset(subject_word3)
;
; Get the first newsgroup listed in the Newsgroup: line, add a meta-word
; u:newsgroup:this-newgroup. Add a link to cache page for a search
; on this meta word
;
newsgroups = DESCRIPTION
newsgroups = /(.\|\n)*?Newsgroups:([\^\n]+\)\n(.\|\n)*$/2/
newsgroups = /\s*(\w|\.)+.*$/1/
addMetaWord(newsgroups)
link_news = "u:newsgroups:"
link_news .= newsgroups
link_news .= ",Newsgroup: "
link_news .= newsgroups
addKeywordLink(link_news)
unset(link_news)
;
```

```
link_thread := subject
addMetaWord(thread)
unset(newsgroups)
link_thread = "u:thread:"
link_thread := thread
link_thread := ",Current Thread"
addKeywordLink(link_thread)
unset(subject)
unset(thread)
unset(link_thread)
```

As a last example of page rules, suppose we wanted to crawl the web and whenever we detected a page had an address we wanted to write that address as a SQL insert statement to a series of text files. We can do this using page rules and the AddressesPlugin. First, we would check the AddressesPlugin and then we might use page rules like:

```
summary = ADDRESSES
setStack(summary)
pushStack(DESCRIPTION)
pushStack(TITLE)
setOutputFolder(/Applications/MAMP/htdocs/crawls/data)
setOutputFormat(sql)
setOutputTable(SUMMARY);
writeOutput(summary)
```

The first line says copy the contents of the ADDRESSES field of the page into a new summary field. The next line says use the summary field as the current stack. At this point the stack would be an array with all the addresses found on the given page. So you could use the command like `popStack(first_address)` to copy the first address in this array over to a new variable `first_address`. In the above case what we do instead is push the contents of the DESCRIPTION field onto the top of the stack. Then we push the contents of the TITLE field. The line

```
setOutputFolder(/Applications/MAMP/htdocs/crawls/data)
```

sets `/Applications/MAMP/htdocs/crawls/data` as the folder that any auxiliary output from the page_processor should go to. `setOutputFormat(sql)` says we want to output sql, the other possibility is csv. The line `setOutputTable(SUMMARY);` says the table name to use for INSERT statements should be called SUMMARY. Finally, the line `writeOutput(summary)` would use the contents of the array entries of the summary field as the column values for an INSERT statement into the SUMMARY table. This writes a line to the file `data.txt` in `/Applications/MAMP/htdocs/crawls/data`. If `data.txt` exceeds 10MB, it is compressed into a file `data.txt.0.gz` and a new `data.txt` file is started.

Search Time Tab

The Page Options Search Time tab looks like:



Word Suggest <input checked="" type="checkbox"/>	Similar <input checked="" type="checkbox"/>
Subsearch <input checked="" type="checkbox"/>	Inlinks <input checked="" type="checkbox"/>
Signin <input checked="" type="checkbox"/>	IP Address <input checked="" type="checkbox"/>
Cache <input checked="" type="checkbox"/>	Result Score <input checked="" type="checkbox"/>

Search Ranking Factors [?](#)

Title Weight:

Description Weight:

Link Weight:

Search Results Grouping [?](#)

Minimum Results to Group:

The Search Page Elements and Links control group is used to tell which elements and links you would like to have presented on the search landing and search results pages. The Word Suggest checkbox controls whether a dropdown of word suggestions should be presented by Yioop when a user starts typing in the Search box. It also controls whether spelling corrections will appear. The Subsearch checkbox controls whether the links for Image, Video, and News search appear in the top bar of Yioop You can actually configure what these links are in the [Search Sources](#) activity. The checkbox here is a global setting for displaying them or not. In addition, if this is unchecked then the hourly activity of downloading any RSS media sources for the News subsearch will be turned off. The Signin checkbox controls whether to display the link to the page for users to sign in to Yioop The Cache checkbox toggles whether a link to the cache of a search item should be displayed as part of each search result. The Similar checkbox toggles whether a link to similar search items should be displayed as part of each search result. The Inlinks checkbox toggles whether a link for inlinks to a search item should be displayed as part of each search result. The IP address checkbox toggles whether a link for pages with the same ip address should be displayed as part of each search result. Finally, the Result Score checkbox controls whether the search result score used for ranking a particular search result item is shown. The Search Ranking Factors group of controls: Title Weight, Description Weight, Link Weight field are used by Yioop to decide how to weigh each portion of a document when it returns query results to you.

When Yioop ranks search results it search out in its postings list until it finds a certain number of qualifying documents. It then sorts these by their score, returning usually the top 10 results. In a multi-queue-server setting the query is simultaneously asked by the name server machine of each of the queue server machines and the results are aggregated. The Search Results Grouping controls allow you to affect this behavior. Minimum Results to Group controls the number of results the name server want to have before sorting of results is done. When the name server request documents from each queue server, it requests for $\alpha * (\text{Minimum Results to Group}) / (\text{Number of Queue Servers})$ documents. Server Alpha controls the number alpha.

The Save button of course saves any changes you make on this form.

Test Options Tab

The Page Options Test Options tab looks like:



Method of Submission ▾

The Method of Submission drop down allows a user to select from one of three methods to test how Yioops page processors would process a web page: By URI, By File Upload, or By Direct Input. Selecting the first method, reveals a form to enter a URI (aka the URL or link to the web page) and a submit button to cause Yioop to download and process that page. The By File Upload method similarly reveals a form when a user can either drag or select a file to process. Either of these two methods is useful for test web pages, images, or any other document Yioop can process. The final method is more geared toward text documents. If one selects By Direct Input, the following form is revealed:

Crawl Time Search Time Test Options

Test Page ?

By Direct Input ▾

Type: text/html ▾

```
<!DOCTYPE html>
<html lang="en-US" dir="ltr" data-redirect="true" data-base="/">
<head>
<title>Open Source Search Engine Software - Seekquarry</title>
<meta name="description"
content="SeekQuarry provides open source search technologies" />
<meta name="Author" content="Chris Pollett" />
<meta name="referrer" content="strict-origin-when-cross-origin" />
<meta charset="utf-8" />
<link rel="icon" href="/wd/resources/favicon.ico" />
<link rel="stylesheet" type="text/css"
href="/css/search.css" />
<link rel="stylesheet" type="text/css"
href="/wd/css/auxiliary.css" />
<style>
    body
    {
        background-color: #e0e0e0;
    }
    .frame,
    .icon-upload,
    .current-activity,
    .light-content,
    .small-margin-current-activity,
    .suggest-list li span.unselected
    {
        background-color: #ffffff;
    }
    .foreground-color,
    .icon-upload
    {
        color: #ffffff;
    }

```

Test Process Page

In the Type dropdown one can select a [MIME Type](#) used to select the page processor Yioop uses to extract text from the data you type or paste into the textarea on this page. For any of the three methods of input, after clicking Test Process Page, the Test Options let's you see how Yioop would process a web page and add summary data to its index. In particular, it will show the \$summary associative array Yioop would create from the page after the appropriate page processor is applied. Beneath this, it shows the \$summary array that would result after user-defined page rules from the crawl time tab are applied. Yioop stores a serialized form of this array in a IndexArchiveBundle for a crawl. Beneath this array, the results will show an array of terms (or character n-grams) that were extracted from the page together with their positions in the document. Next, a list of meta words that the document has are listed. Either extracted terms or meta-word could be used to look up this document in a Yioop index. Finally, the times to carry out the above processing will also be shown.

[Return to table of contents](#) .

Web Scrapers

Web sites are often constructed using some kind of content management system (CMS) such as [Drupal](#) or [WordPress](#). Such web site typically have their non-navigational content at the same location on a page across all their pages. For example, it may appear in the same div tag across all their pages. In terms of improving the quality of search results, it

information for a page, etc. So it is useful to have a means to detect that such meta information exists and to appropriately scrape it. The Web Scraper activity allows one to configure content scrapers for particular CMS's, versions of particular CMS's, or for particular meta information formats such as [Open Graph](#). Web Scrapers make use of [XPath](#), so to use Web Scrapers effectively one should become familiar with XPath. The Web Scrapers activity looks like:



Name:

Signature:

Priority:

Text XPath:

Delete XPaths:

Extract Fields:

Save

Scrapper Info	Actions	
DRUPAL Signature: /html/head/*[contains(@href, '/sites/all/themes') or contains(@href, '/sites/de		
Priority: 0		
Text XPath: //div[@id='page']//main	Edit	Delete
Delete XPaths: /**[contains(@id,'comments')] /**[contains(@id,'respond')] /**[contains(@class,'bottomcontainerBox')] /**[contains(@class,'post-by')] /**[contains(@class,'entry meta-clear')]		
Extract Fields:		
MEDIAWIKI Signature: //meta[contains(@content, 'MediaWiki')]		
Priority: 0		
Text XPath: /**[contains(@id, 'mw-content-text')]	Edit	Delete
Delete XPaths: /**[contains(@class, 'nmbbox')] /**[contains(@class, 'hatnote')] /**[contains(@class, 'infobox')]		
Extract Fields:		
VBULLETIN Signature: /html/head/*[contains(@href, 'vbulletin')]		
Priority: 0		
Text XPath: //div[contains(@class, 'body_wrapper')]		
Delete XPaths: /**[contains(@id, 'above')] /**[contains(@id, 'below')] /**[contains(@id, 'breadcrumb')] /**[contains(@id, 'notices')] /**[contains(@id, 'footer')] /**[contains(@id, 'forum_info_options')] /**[contains(@class, 'threadlisthead')] /**[contains(@class, 'threaddetails')] /**[contains(@id, 'pagination')] /**[contains(@class, 'threadstats')] /**[contains(@class, 'threadlastpost')] //span[contains(@class, 'label')]	Edit	Delete
Extract Fields:		
VIDEO SITE Signature: //meta[@property='og:type' and contains(@content, 'video')]		
Priority: 0		

**Extract Fields:**

```
IS_VIDEO>//meta[@property='og:type' and contains(@content, 'video')]/@content
IS_VR>//meta[(@property='og:video:tag' (contains(@content, '360') or contains(
SITE_NAME>//meta[@property='og:site_name']/@content
DURATION>//meta[@property='video:duration']/@content
THUMB_URL>//meta[@property='og:image']/@content
```

The **Add Scraper** form at the top of the activity can be used to add new web scrapers. The list of current scrapers appears below this and can be used to edit, delete, and manage existing scrapers. On the **Add Scraper** form or **Edit Scraper** form, there is a **Name** field. **Name** is what to call the scraper that is being defined. A Web Scraper must have a Name, the Signature and Scrape Rules fields are optional but at least one of them must be present for the web scraper to have effect while crawling.

Signature is used to detect when a particular Web Scraper should be used. It should consist of an XPath query which would evaluate to a non-empty set of elements in the case of a page the scraper might work for.

Priority is used to specify the priority of this particular scrape rule. A rule with a higher priority will be applied before one of lower priority.

Text XPath is used to specify an XPath to the most important content of a page for summarization.

Delete XPaths is used to specify XPaths, one per line, of content under the Text XPath portion of the web page, that should be non considered for summarizations.

Extract Fields is used to specify a sequence of rules to extract to specific fields in the summary. Each rule should be on a line by itself and have the format: NAME_OF_SUMMARY_FIELD = SOME_XPATH. Such a rule computes the XPath on the original document and concatenates the text contents of the resulting nodes into NAME_OF_SUMMARY_FIELD in the summary. For example,

```
SITE_NAME>//meta[@property='og:site_name']/@content
```

would take the value of the content attribute of all meta tags with property attribute having value og:site_name, concatenate them as a string, and store the key SITE_NAME with value this string in the pages summary when it is indexed. As can be seen in the image of the Web Scraper Activity, Extract Fields are used to determine if a page has a video on it provided the page uses open graph.

[Return to table of contents](#) .

Results Editor

Sometimes after a large crawl one finds that there are some results that appear that one does not want in the crawl or that the summary for some result is lacking. Sometimes also a user just wants a quick summary of a topic without needing to go to a separate web page. The Result Editor activity allows one to address these issues on search results pages. The Results Editor activity looks like:



Edit Results Query Map Knowledge Wiki

Edit Result ?

URL: Load

As one can see it has three tabs: Edit Results, Query Map, and Knowledge Wiki. The first tab shows by default and allows a user to see what the default search results for a url would look like. One enters the url and it returns a page such as the below:

Query

Top Urls For Query

Back

Save

The textarea **Top Urls for Query** then allows one to enter urls, one per line, that one wants at the top of the search results page for that query. Results beyond those urls listed will be computed as normal from the remaining search results in the index for that query. Search results that have been output using a Query Map would look like:

The last tab of the Results Editor activity let's one configure knowledge wiki callouts that appear on the opposite side of search result pages. These callouts as mentioned above are intended to provide a quick summary of a topic without needing to go to a separate web page.

Below is what the Knowledge Wiki Tab looks like after a query has been entered and any previously configured wiki entry for that query has been loaded:



Query Barack Obama

Knowledge Wiki Page

B *I* U ~~S~~                                      



direct which media sources are present. Clicking the media sources tab's add icon, reveals the **Add Media Source Form** :

Add Media Source ?

Type:

Name:

URL:

Language:

Category:

Provide xpath to feed components below:

Image XPath:

Trending Stop Term Regex:

This form is used to add and edit media sources used by Yioop, the bottom of the activity is used to manage different kinds of subsearches available in the particular Yioop instance. The top portion is somewhat more involved, so will take a fair bit of explanation before we get to the subsearch portion of the activity. For Media Sources, the Media Kind can be either RSS, HTML, JSON, Regex, Feed Podcast, Scrape Podcast, or Trending value. We next describe each of these in turn and how the rest of the Add/Edit form works for them. before we get started we mention the last field on the above form, **Trending Stop Term Regex**, which can usually be left blank. All the media sources except the last three are updated using the FeedUpdateJob which also computes trending terms based on feed items. Sometimes for a feed these items have a term which one wants to exclude from the trending term computation. For example, Daily Beast occurs frequently in Yahoo News items but is not particularly relevant to a feed item. **Trending Stop Term Regex** can be used to give a regex such as `/Daily(\s|-)+Beast/i` to remove any such words from the trending computation. We now return to discussing the media source possibilities and omit mention of this field for the remainder of the discussion.

An **RSS media source** can be used to add an RSS or Atom feed (it auto-detects which kind) to the list of feeds which are downloaded hourly when Yioop's Media Updater is turned on. Besides the name you need to specify the URL of the feed in question. The Category field search usually be left at news. If you want to specify additional categories such as weather or sports, you typically want to create a mix that searches the default index with the keyword `media:your_category` injects, and then make a new subsearch with that mix. This will allow your new category to show up on the Tools/More/Other Searches page.

An **HTML media source** is a web page that has feed articles like an RSS page that you want the Media Updater to scrape on an hourly basis. To specify where in the HTML page the news items appear you specify different XPath information. For example,

```
Name: Cape Breton Post
URL: http://www.capebretonpost.com/News/Local-1968
Language: English
Category: news
Channel: //div[contains(@class, "channel")]
Item: //article
Title: //a
Description: //div[contains(@class, "dek")]
Link: //a
```

The Channel field is used to specify the tag that encloses all the news items. Relative to this as the root tag, `//article` says the path to an individual news item. Then relative to an individual news item, `//a` gets the title, etc. Link extracts the href attribute of that same `//a`.

A **JSON media source** is used to scrape feed articles from JSON data as may be provided by a websites API. To handle a JSON media source you provide the same information as with an HTML media source. Internally, Yioop converts all JSON sources to xml before processing. The root objects maps to `/html/body`. A property *foo* of the root



are used to extract the feed items in the same way as how HTML feeds are extracted. For an example, Yioop search results and discussion groups can be output as JSON. To take Yioop's news feed and use it as a JSON media source in your search engine, you could use the settings:

```
Name: Yioop News
URL: https://www.yioop.com/s/news?f=json
Language: English
Category: news
Channel: //channel
Item: //item
Title: //title
Description: //description
Link: //link
```

A **Regex media source** is a source of feed articles presented in some kind of non-tag based text format. For example, the US National Weather Service has a text-based page for weather forecasts of major US cities at

```
http://forecast.weather.gov/product.php?site=NWS&
issuedby=04&product=SCS&format=txt&
version=1&glossary=0
```

changing the 04 above to 03, 02, 01 varies the group of cities. Most of the data on this page appears in a pre tag as text. *Channel* in this case would be a regex whose first capture group corresponds to the contents of this pre tag. We might want to get one item per line from the pre tag as that would correspond to the weather for one city. The *Item Separator* is a regex used to split the results of the Channel operation into items. Finally, *Title*, *Description*, and *Link* are regexes each with one capture group used to get these respective feed item components out of an item given after the splitting process above. Hence, a reasonable choice of values for the weather service page might be:

```
Name: National Weather Service 04
URL: http://forecast.weather.gov/product.php?
site=NWS&issuedby=04&product=SCS&format=txt&
version=1&glossary=0
Language: English
Category: weather
Channel: /<pre(?:..+?)>([^\<]+)/m
Item: /\n/
Title: /^(.+?)\s\s\s+/
Description: /\s\s\s+(.+?)$/
Link: http://www.weather.gov/
```

Notice in the above that the Link element is <http://www.weather.gov/>. If you have a feed and it doesn't provide links for individual items. You can always provide a link to some fixed site by directly entering a URL in the Link field.

Not all feeds use the same tag to specify the image associated with a news item. The Image XPath allows you to specify relative to a news item (either RSS or HTML) where an image thumbnail exists. If a site does not use such thumbnail one can prefix the path with ^ to give the path relative to the root of the whole file to where a thumb nail for the news source exists. Yioop automatically removes escaping from RSS containing escaped HTML when computing this. For example, the following works for the feed:

```
https://feeds.wired.com/wired/index
//description/div[contains(@class, "rss_thumbnail")]/img/@src
```

A **Feed Podcast source** is an RSS or Atom source where each item contains a link to a podcast or video podcast. For example,

```
http://feed.cnet.com/feed/podcast/all/hd.xml
```

The **Alternative Link Tag** field is used to say the XPath within the feed item to the link for the audio or video file. For the CNet example, this is:

```
enclosure
```

If it is blank the default link tag is used. The media updater job when run checks if any



Expires field controls how long a feed item is kept before it is deleted. For example, if we wanted to download the popular Ted talk podcasts into the Ted subfolder of the resource folder of the Example Podcast wiki page of the Public group, where we have podcasts expire after after 1 month, we could do:

```
Name: Ted
URL: https://pa.tedcdn.com/feeds/talks.rss
Language: English
Expires: One Month
Alternative Link Tag: enclosure
Wiki Destination: Library@News and Podcasts/Ted/%Y-%m-%d %F
```

Notice the string has "%Y-%m-%d %F" in it. This portion of the destination gives the format of the filename to use when storing a downloaded podcast file. It says name the file as the current year hyphen month hyphen day space the filename as given in the URL. %F is for the filename, other % modifiers can be standard date formatting instructions.

Yioop supports the downloading of single video or audio file sources, as well as more complicated stream sources such as m3u8 streams.

A **Scrape podcast source** is like a **Feed Podcast source**, but where one has a HTML or XML page which has a periodically updated link to a video or audio source. For example, it might be an evening news web site. The URL field should be the page with the periodically updated link. The **Aux Url XPaths** field, if not blank, should be a sequence of XPaths or regexes one per line. The first line will be applied to the page to obtain a next url to download. The next line's XPath or regex is applied to this file and so on. The final url generated should be to the HTML or XML page that contains the media source for that day. Finally, on the page for the given day, **Download XPath** should be the XPath of the url of the video or audio file to download. If a regex is used rather than an XPath, then the first capture group of the regex should give the url. A regex can be followed by `json|` to indicate the first capture group should be converted to a json object. To reference a path of through sub-objects of this object to a url. As an example, consider the following, which at some point, could download the Daily News Scrape Podcast to a wiki group:

```
Type: Scrape Podcast
Name: Dailly News Podcast
URL: https://www.somenetwork.com/daily-news
Language: English
Aux Url XPaths:
/(https:\\\\cdn.somenetwork.com\\daily-news\\video\\daily-[""]+)"
/window\\.\\_\\_data\\s*\\=\\s*{ [""]+[""]+\\};)/json|video|current|0|publicUrl
Download XPath: //video[contains(@height,'540')]
Wiki Destination: My Private Group@Podcasts/%Y-%m-%d.mp4
```

The initial page to be download will be: <https://www.somenetwork.com/daily-news>. On this page, we will use the first Aux Path to find a string in the page that matches `/(https:\\\\cdn.somenetwork.com\\daily-news\\video\\daily-[""]+)"`. The contents matching between the parentheses is the first capture group and will be the next url to download. SO for example, one might get a url:

```
https://cdn.somenetwork.com/daily-news/video/daily-safghdsjfg
```

This url is then downloaded and a string matching the pattern `/window\\.__data\\s*\\=\\s*{ [""]+[""]+\\};)` is found. The capture group portion of this string consists of what matches `[""]+[""]+\\};)` is then converted to a JSON object, because of the `json|` in the Aux Url XPath. From this JSON object, we look at the video field, then the current subfields, its 0 subfield, and finally, the `publicUrl` field. This is the url we download next. Lastly, the download XPath is then used to actually get the final video link from this downloaded page. Once this video is downloaded, it is stored in the Podcasts page's resource folder of the the My Private Group wiki group in a file with a name in the format: `%Y-%m-%d.mp4`.

A **Trending value source** is a value on a web page that one would like to track using Yioop's trending search mechanism. The Name field is the name to use for the trending value. The URL field should be the page with the periodically updated value. **Category** should be the trends category (a collection of trending values) one would like to track this value with. **Group Within Category** is the default name of the key that will be associated



more capture groups, the first two capture groups are used to give a key name, value pair for a particular download time. As an example,

```
Name: Yioop Ticker
URL: https://my-great-stock-quotes/yioop
Language: English
Category: stocks
Group Within Category: Yioop Price
Trend Value Regex: /Yioop\:\s+(\d+\.\d+)/
```

Here there is only one capture group (\d+\.\d+), so searching on trending:stocks, one would see all the hour, weekly, etc values for the trending values with that category. One such row would be Yioop Price whose values would be computed based on the numbers extracted according to this regex's (\d+\.\d+) capture group.

The second tab on the activity manages subsearches associated with this instance of Yioop:

Folder Name	Source:	Localization Identifier	Results per Page	Default Query and Format	Actions
images	Mix:images m:2	db_subsearch_images	50	Query: Landing: No Highlight	Edit Localize Delete
news	Mix:news m:4	db_subsearch_news	20	Query: lang:default-major Landing: 1	Edit Localize Delete
trends	Trend Category: news	db_subsearch_trends	N/A	Sort: score desc Landing: 2	Edit Localize Delete
videos	Mix:videos m:3	db_subsearch_videos	10	Query: Landing: No Highlight	Edit Localize Delete

Clicking its add icon reveals the form:

Add a Subsearch ?

Folder Name:

Source:

Results Per Page:

Landing Priority:

Default Query:

This form allows you to add a new specialized search link which may appear on the Search Context Menu. It can also be used to specify if top five results for a default query of this subsearch should appear on the Yioop landing page. The **Add a Subsearch** form has five fields: **Folder Name** is a short familiar name for the subsearch, it will appear as part of the query string when the given subsearch is being performed. For example, if the folder names was news, then s=news or /s/news (if redirects are on) will appear as part of the query string when a news subsearch is being done. Folder Name is also used to make the localization identifier used in translating the subsearch's name into different languages. This identifier will have the format db_subsearch_identifier. For example, db_subsearch_news. **Index Source**, the second form element, is used to specify a crawl or a crawl mix that the given subsearch should use in returning results. It also has a value **Trend Category**. If this is chosen you can make a subsearch based on trending terms for a media source category such as news. The **Results per Page** form element controls the number of search results which should appear when using this kind of subsearch. The **Landing Priority** form element controls whether top five results of the **Default Query** in the next field, or for trends nine randomly chosen trending values, should be displayed on the landing page. If this dropdown is set to any value other than **No Highlight**, these results will be displayed. The order of displaying the different subsearch is then based on the priority given in this dropdown (higher means earlier).



and that particular subsearch localization identifier, so that you can fill in a value for it. Remembering the name of this identifier, one can then in Manage Locales navigate to other locales, and fill in translations for them as well, if desired.

[Return to table of contents](#) .

GUI for Managing Machines and Servers

Rather than use the command line as described in the [Prerequisites for Crawling](#) section, it is possible to start/stop and view the log files of queue servers and fetcher through the Manage Machines activity. This activity also is used to turn on and off the Media Updater process/processes. These processes are responsible for updating news feeds, new group and thread recommendations, trending terms and values, sending out bulk e-mail notifications about group activities, and if run in a distributed mode, converting videos to mp4 format. In order for Manage Machines to work, the additional requirements for this activity mentioned in the [Requirements](#) section must have been met. The Manage Machines activity looks like:

Machine Statuses

[\[Manage Crawls\]](#)

Media Updater

[\[Configure Media Jobs\]](#)

Name Server

Media Updater [\[Log\]](#) On [Off](#)

Row 0 to 4 of 4 Show

Machines ⊞

FG1 [Channel:0][https://173.13.143.76:8000/] [\[Delete\]](#)

Queue Server #00 [\[Log\]](#)

On [Off](#)

Fetchers

#00 [\[Log\]](#)
On [Off](#)

#01 [\[Log\]](#)
On [Off](#)

#02 [\[Log\]](#)
On [Off](#)

#03 [\[Log\]](#)
On [Off](#)

#04 [\[Log\]](#)
On [Off](#)

#05 [\[Log\]](#)
On [Off](#)

FG2 [Channel:0][https://173.13.143.76:8001/] [\[Delete\]](#)

Queue Server #00 [\[Log\]](#)

Since Manage Machines is frequently used in conjunction with crawling, we see under the Machine Statuses heading a link to Manage Crawls. The Media Updater section has a link to a separate page where one can choose which jobs will be run by the Media Updater and how they will be run: I.e., in a distributed or non distributed fashion. If the Media Updater is configured to be non-distributed (default), that is to only run on the name server, then beneath this link there will be an on off toggle that let's one turn on and off the Media Updater. If it is off, feeds, trends, analytic counts, etc will not be periodically updated, old data though will persist. The page the configure link takes one to looks like:



JOB Name	Run Status
Analytics	<input type="checkbox"/> On
BulkEmail	<input type="checkbox"/> On
FeedsUpdate	<input type="checkbox"/> On
Recommendation	<input type="checkbox"/> Off
TrendingHighlights	<input type="checkbox"/> On
VideoConvert	<input type="checkbox"/> On
WikiMedia	<input type="checkbox"/> On

The mode toggle allows one to switch between distributed and non-distributed mode for the Media Updater. Distributed mode can handle more feeds, etc., however, it is more complicated and prone to crashes as the media updater needs to be run separately on each machine in the Yioop instance, and any crash amongst these can prevent the updater as it is currently written from working properly. Beneath the mode toggle on this page are on/off toggle for each of the available jobs for the media updater. Descriptions of each job can be read in the integrated help. The subsection on **Writing a MediaJob** in [Advanced Topics](#) describes how to add new jobs of your own design. Referring back to the Manage Machines Activity, beneath the Media Updater section is the Machines table. The table itself lists the states of the currently available machines. Clicking the add icon in its heading, reveals a form to add a new machine.

Add Machine

Name:

URL:

Channel Type: 

Number of Fetchers: 

On the **Add Machine** form, the **Machine Name** field lets you give this machine an easy to remember name. The **Machine URL** field should be filled in with the URL to the installed Yioop instance. The **Type/Channel** dropdown lets one either select Replica Server as the type or a numeric number as a channel. A Replica Server is used to make a mirror copy of an existing Queue Server. This mirror is only used when computing query results, but can be used to speed up computing results if a particular machine is getting too swamped with requests. If one selects Replica Server a dropdown, Parent Name, is revealed where the user can select which machine is being replicated. If rather than select **Replica Server**, one selects a numeric channel, then a new machine with a Queue Server and Fetchers will be created when the form is submitted. This new machine will be assigned to the selected channel. When a user runs a crawl, only crawls given the same channel number as this new machine will make use of it. The channel mechanism is what Yioop uses to allow for more than one simultaneous crawl. To complete our description of the **Add Machine** form, the **Number of Fetchers** dropdown allows you to say how many fetcher instances you want to be able to manage for that machine. All machines other than Replica Server will have a Queue Server, but they may have a variable number of Fetchers.

The Machines table itself consists of a set of boxes for each machine you have added to Yioop. In the far corner of this box is a link to Delete that machine from the list of known machines, if desired. Besides this, each box lists the queue server, if any, and each of the fetchers you requested to be able to manage on that machine. Next to these there is a link to the log file for that server/fetcher and below this there is an On/Off switch for starting and stopping the server/fetcher. This switch is green if the server/fetcher is running and red



introduction.

Analytics in Yioop

A Yioop instance may collect search query, thread view, wiki page view, and group view data depending on how it is configured. It may also collect data on whether an advertisement link was clicked or not. Search query and view statistics are available to admin accounts and group owner to help them understand what is popular on their site. Thread view, wiki page view, and group view information is used to help populate a user's list of recently viewed threads, groups, and pages in the feed and wiki navigation dropdown. Thread views are also displayed when a user is looking at threads in various group feed pages.

Configuration and Updating

The **Security Activity** described in the [Optional Server and SecurityConfigurations](#) has several dropdowns related to analytics. The **Search Analytics** and **Group Analytics** dropdowns can be used to respectively turn on or off search query statistics and feed, wiki, and group statistics. The **Differential Privacy** dropdown controls whether or not statistics displayed to end users for things like thread views are fuzzified or not. If they are fuzzified, then gaussian noise is added to view counts, so that it is hard for someone to glean personal information from any aggregate counts or averages displayed.

Aggregate statistics such as number of hourly, daily, etc views are computed by the code in `src/library/media_jobs/AnalyticsJob.php`, which is run by the MediaUpdater process. This process can be turned on and off under the **Manage Machines** activity. If this job is not running, then these statistics are not updated.

Search Analytics

As the root user a link for query statistics shows up on a user's main dashboard page, and also appears next to Crawls table header under the **Manage Crawls** activity.

Crawls and Indexes

Crawl and index the web or an existing archive and create a searchable index. You have 1 active crawls, 3 previous crawl indexes.

[\[Manage Crawls and Indexes\]](#) [\[Search Query Statistics\]](#) [\[Manage Crawl Machines\]](#)

Crawls [\[Query Statistics\]](#)

Clicking on these links takes one to a page showing recent queries and the number of times they have been queried:

Last Hour		Last Day		Last Month		Last Year	
	Views		Views		Views		
lang:default-major highlight:3	5	lang:default-major highlight:3	78	lang:default-major highlight:3	2629	lang:default-major	
tuto lang:fr	2	呂主席	8	lang:default-major highlight:1	107	lang:default-major highlight:3	
Go	2	lang:default-major highlight:1	6	x lang:fr	63	lang:default-major highlight:1	
Web search	2	do lang:fr	6	do lang:fr	63	COVID-19	
août lang:fr	1	COVID-19	5	60 lang:fr	54	lang:fr	
barcelone lang:fr	1	mac lang:fr	4	35 lang:fr	51		

The Filter form can be used to narrow down the keywords one is looking for statistics on -- only. queries matching the filter terms will be displayed.

Groups Analytics

Owner's of groups can see statistics about how users are accessing threads and wiki pages within a group. A statistics link that leads to this information can be found both on the

Last Post:-- why the YIOOP Home Page does not display trends, news and sports in the default Search Engine Language ???

Clicking on this link takes one to the following Statistics Page:

Yioop Software Help Group Statistics

Filter:

Group Views

Last Hour	No Activity
Last Day	110
Last Month	2472
Last Year	24420
All Time	194181

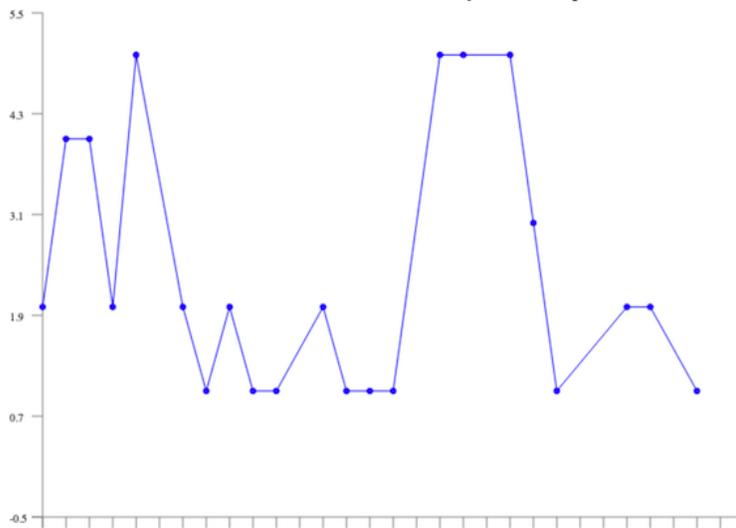
Thread Views

Last Hour	Views	Last Day	Views	Last Month	Views	Last Year
No Activity		Crawler Halting	4	Machine don't initialize	55	Machine don't initialize
		how ... ??? - Queue Server not work in my server vps	4	indexing problem	53	how ... ??? - Queue Server not work in my server vps
		setup problems	3	Yioop Version 2.0 Released!	47	Crawl doesn't work
		yioop plugin post processing	3	setup failing on centos server	46	

Links on a statistics page take one to charts for that item:

Yioop Software Help Thread Views : Last Month

Visits per Day



Day	Number of Visits
07-23-2020	2
07-24-2020	4
07-25-2020	4
07-26-2020	2

Building Sites with Yioop

Building a Site using Yioop's Wiki System

As was mentioned in the [Appearance Activity](#) section of the documentation, background color, icons, title, and SEO meta information for a Yioop instance call all be configured from



the form on the [Server Settings](#) activity. If you would like a site with a more custom landing page, then one can check **Use Wiki Public Main Page as Landing Page** under Toggle Advance Settings : Site Customizations. The Public Main page will then be the page you see when you first go to your site. You can then build out your site using the wiki system for the public group. Common headers and footers can be specified for pages on your site using each wiki page's Settings attributes. More advanced styling of pages can be done by specifying the auxiliary css data under Toggle Advance Settings. As Wiki pages can be set to be galleries, or slide presentations, and as Yioop supports including images, video, and embedding search bars and math equations on pages using its [Yioop's Wiki Syntax](#) , one can develop quite advanced sites using just this approach. The video tutorial [Building Websites Using Yioop](#) explains how the Seekquarry.com site was built using Yioop Software in this way.

Building a Site using Yioop as Framework

For more advanced, dynamic websites than the wiki approach described above, the Yioop code base can still serve as the code base for new custom search web sites. The web-app portion of Yioop uses a [model-view-adapter \(MVA\) framework](#). This is a common, web-suitable variant on the more well-known Model View Controller design pattern. In this set-up, sub-classes of the Model class should handle file I/O and database function, sub-classes of Views should be responsible for rendering outputs, and sub-classes of the Controller class do calculations on data received from the web and from the models to give the views the data they finally need to render. In the remainder of this section we describe how this framework is implemented in Yioop and how to add code to the WORK_DIRECTORY/app folder to customize things for your site. In this discussion we will use YIOOP_DIR to refer to the Yioop folder, APP_DIR to refer to WORK_DIRECTORY/app and BASE_DIR to refer to YIOOP_DIR/src.

To understand Yioop's MVA architecture, we begin by tracing how a typical request is processed by Yioop. A typical web app request is handled by the file YIOOP_DIR/index.php. It defines a constant seekquarry\yioop\configs\REDIRECTS_ON as true and then loads the file BASE_DIR/index.php whose bootstrap() method is then called. Since seekquarry\yioop\configs\REDIRECTS_ON is true, bootstrap() will call a function configureRewrites() to see if the incoming URL should be rewritten. To check for rewrite, configureRewrites() makes use of an associative array \$route_maps consisting of incoming_route => routeHandlerFunctionName pairs. For example, the pair 'admin' => 'routeController' says that an incoming url of the form YIOOP_LOCATION/admin should be rewritten by calling the function routeController. Before performing a rewrite lookup, configureRewrites() checks if a static method Routes::getRoutes() exists, and if so, tacks on the associate array this method returns to \$route_maps. So if you want to add your own rewrite you can define the getRoutes method in a Routes class in a file APP_DIR/Routes.php. Typically, when one defines a handler function for a given route, one also defines a helper function, that let's one create urls for links which match the given pattern. For example, function controllerUrl(\$name, \$with_delim = false) can be used to create urls which could be handled routeController. This function should check if REDIRECTS_ON is true and output a url that works for Yioop if url rewriting is on and a different one if it is off.

Once url rewriting has been performed, bootstrap() tries to determine which controller should be used for the incoming request. To do this bootstrap() has an array \$available_controllers which lists the controllers available to the script. The names of the controllers in this array are lowercase. Based on whether the \$_REQUEST['c'] variable is in this array index.php either loads the class upperCaseFirstLetter(\$_REQUEST['c'])Controller or loads whatever the default controller is. bootstrap() completes by instantiating this class and calling its processRequest() method. To add to the list of \$available_controllers, a developer can create function localControllers in their BASE_DIR/configs/LocalConfig.php which returns an array of additional controller names. Yioop's autoloader first check the directory APP_DIR/controllers for the existance of a controller and if this fails looks in the



one should have in it a file which extends the class Controller. Controller files should always have names of the form SomenameController.php and the class inside them should be named SomenameController. Notice it is Somename rather than SomeName. These general naming conventions are used for models, views, etc. Any Controller subclass has methods component(\$name), model(\$name), view(\$name), and plugin(\$name). These methods load, instantiate, and return a class with the given name. For example, `$my_controller->model("crawl");` checks to see if a CrawlModel has already been instantiated, if so, it returns it; if not, it tries to load it, first from the file `APP_DIR/models/CrawlModel.php`, and if that fails, from `BASE_DIR/models/CrawlModel.php`. Once loaded, it instantiates a CrawlModel, saves a reference to this object, and returns it.

As we indicated above, if a class needs to be loaded, Yioop first looks in `APP_DIR`. For example, `$my_controller->view("search")` would first look for a file: `APP_DIR/views/SearchView.php` to include, if it cannot find such a file then it tries to include `BASE_DIR/views/SearchView.php`. So to change the behavior of an existing `BASE_DIR` file one just has a modified copy of the file in the appropriate place in your `APP_DIR`. This holds in general for other program files such as components, models, and plugins. It doesn't hold for resources such as images -- we'll discuss those in a moment. Notice because it looks in `APP_DIR` first, you can go ahead and create new controllers, models, views, etc which don't exist in `BASE_DIR` and get Yioop to load them.

A Controller must implement the abstract method `processRequest`. Recall this is the method that `bootstrap()` will call. The `processRequest` method should make use of data gotten out of the loaded models as well as data from the web request to do some calculations. Typically, to determine the calculation performed, the controller cleans and looks at `$_REQUEST['a']`, the request activity, and uses the method `call($activity)` to call a method that can handle the activity. When a controller is constructed it makes use of its static variable `$component_activities` to know which components have been associated with it and what activities from these components are allowed. The `call()` method checks if there is a Component responsible for the requested activity, if there is, it calls that Component's `$activity` method, otherwise, the method that handles `$activity` is assumed to come from the controller itself. The results of the calculations done in `$activity` would typically be put into an associative array `$data`. After the `call()` method completes, `processRequest` typically takes `$data` and calls the base Controller method `displayView($view, $data)`. Here `$view` is the whichever loaded view object you would like to display.

To complete the picture of how Yioop eventually produces a web page or other output, we now describe how subclasses of the View class work. Subclasses of View have a field `$layout` and two methods `helper($name)`, and `element($name)`. A subclass of View has at most one Layout and it is used for rendering the header and footer of the page. It is included and instantiated by setting `$layout` to be the name of the layout one wants to load. For example, `$layout="web";` would load either the file `APP_DIR/views/layouts/WebLayout.php` or `BASE_DIR/views/layouts/WebLayout.php`. This file is expected to have in it a class `WebLayout` extending `Layout`. The constructor of a `Layout` takes as argument a view which it sets to an instance variable. The way `Layout`'s get drawn is as follows: When the controller calls `displayView($view, $data)`, this method does some initialization and then calls the `render($data)` of the base View class. This in turn calls the `render($data)` method of whatever `Layout` was on the view. This `render` method then draws the header and then calls `$this->view->renderView($data);` to draw the view, and finally draws the footer.

The methods `helper($name)` and `element($name)` of View load and instantiate, if necessary, and return the Helper or Element `$name` in a similar fashion to the `model($name)` method of Controller. Element's have `render($data)` methods and can be used to draw out portions of pages which may be common across Views. Helper's on the other hand are used typically to render UI elements. For example, `OptionsHelper` has a `render($id, $name, $options, $selected)` method and is used to draw select dropdowns.

When rendering a View or Element one often has css, scripts, images, videos, objects, etc.



LINKS/RESOURCE_PATH, doesn't have to be under your web server's DOCUMENT_ROOT. So how does one refer in a link to these folders? To handle this one uses Yioop's ResourceController class which can be invoked by a link like:

```

```

Here c=resource specifies the controller, a=get specifies the activity -- to get a file, n=myicon.png specifies we want the file myicon.png -- the value of n is cleaned to make sure it is a filename before being used, and f=resources specifies the folder -- f is allowed to be one of css, script, or resources. This would get the file APP_DIR/resources/myicon.png .

This completes our description of the Yioop framework and how to build a new site using it. It should be pointed out that code in the APP_DIR can be localized using the same mechanism as in BASE_DIR. More details on this can be found in the section on [Localizing Yioop](#) .

[Return to table of contents](#) .

Embedding Yioop in an Existing Site

One use-case for Yioop is to serve search result for your existing site. There are three common ways to do this: (1) On your site have a web-form or links with your installation of Yioop as their target and let Yioop format the results. (2) Use the same kind of form or links, but request an OpenSearch RSS Response from Yioop and then you format the results and display the results within your site. (3) Your site makes functions calls of the Yioop Search API and gets either PHP arrays or a string back and then does what it wants with the results. For access method (1) and (2) it is possible to have Yioop on an different machine so that it doesn't consume your main web-site's machines resources. As we mentioned in the configuration section it is possible to disable each of these access paths from within the Admin portion of the web-site. This might be useful for instance if you are using access methods (2) or (3) and don't want users to be able to access the Yioop search results via its built in web form. We will now spend a moment to look at each of these access methods in more detail...

Accessing Yioop via a Web Form

A very minimal code snippet for such a form would be:

```
<form method="get" action='YIOOP_LOCATION'>
<input type="hidden" name="its" value="TIMESTAMP_OF_CRAWL_YOU_WANT" />
<input type="hidden" name="l" value="LOCALE_TAG" />
<input type="text" name="q" value="" />
<button type="submit">Search</button>
</form>
```

In the above form, you should change YIOOP_LOCATION to your instance of Yioop's web location, TIMESTAMP_OF_CRAWL_YOU_WANT should be the Unix timestamp that appears in the name of the IndexArchive folder that you want Yioop to serve results from, LOCALE_TAG should be the locale you want results displayed in, for example, en-US for American English. In addition, to embedding this form on some page on your site, you would probably want to change the resources/yioop.png image to something more representative of your site. You might also want to edit the file views/search_view.php to give a link back to your site from the search results.

If you had a form such as above, clicking Search would take you to the URL:

```
YIOOP_LOCATION?its=TIMESTAMP_OF_CRAWL_YOU_WANT&l=LOCALE_TAG&q=QUERY
```

where QUERY was what was typed in the search form. Yioop supports two other kinds of queries: Related sites queries and cache look-up queries. The related query format is:

```
YIOOP_LOCATION?its=TIMESTAMP_OF_CRAWL_YOU_WANT&l=LOCALE_TAG&a=related&garg=URL
```

where URL is the url that you are looking up related URLs for. To do a look up of the Yioop cache of a web page the url format is:

```
YIOOP_LOCATION?its=TIMESTAMP_OF_CRAWL_YOU_WANT&l=LOCALE_TAG&q=QUERY&a=cache&garg=URL
```



The same basic idea as above can return RSS or JSON results simply by appending to the end of the them `&f=rss` or `&f=json`. If you would like to embed json return in a callback Javascript function (JSONP), you can append `&f=json&callback=myCallbackFunction`. This of course only makes sense for usual and related url queries -- cache queries return web-pages not a list of search results. Here is an example of what a portion of an RSS result might look like:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0" xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
xmlns:atom="http://www.w3.org/2005/Atom"
>
  <channel>
    <title>PHP Search Engine - Yioop : art</title>
    <language>en-US</language>
    <link>http://localhost/git/yioop/?f=rss&q=art&its=1317152828</link>
    <description>Search results for: art</description>
    <opensearch:totalResults>1105</opensearch:totalResults>
    <opensearch:startIndex>0</opensearch:startIndex>
    <opensearch:itemsPerPage>10</opensearch:itemsPerPage>
    <atom:link rel="search" type="application/opensearchdescription+xml"
href="http://localhost/git/yioop/yioopbar.xml"/>
    <opensearch:Query role="request" searchTerms="art"/>

    <item>
      <title> An Online Fine Art Gallery U Can Buy Art -
Buy Fine Art Online</title>

      <link>http://www.ucanbuyart.com/</link>
      <description> UCanBuyArt.com is an online art gallery
and dealer designed... art gallery and dealer designed for art
sales of high quality and original... art sales of high quality
and original art from renowned artists. Art</description>
    </item>
    ...
  </channel>
</rss>
```

Notice the `opensearch:` tags tell us the `totalResults`, `startIndex` and `itemsPerPage`. The `opensearch:Query` tag tells us what the search terms were.

Accessing Yioop via the Function API

The last way we will consider to get search results out of Yioop is via its function API. The Yioop Function API consists of the following three methods in `src/controllers/SearchController.php` :

```
function queryRequest($query, $results_per_page, $limit = 0)

function relatedRequest($url, $results_per_page, $limit = 0,
    $scrawl_time = 0)

function cacheRequest($url, $highlight=true, $terms = "",
    $scrawl_time = 0)
```

These methods handle basic queries, related queries, and cache of web page requests respectively. The arguments of the first two are reasonably self-explanatory. The `$highlight` and `$terms` arguments to `cacheRequest` are to specify whether or not you want syntax highlighting of any of the words in the returned cached web-page. If wanted then `$terms` should be a space separated list of terms.

An example script showing what needs to be set-up before invoking these methods as well as how to extract results from what is returned can be found in the file `examples/SearchApi.php`. If you are building a project that uses this API, it might be useful to develop your project using the very convenient PHP package manager known as Composer. Please check out the [Tutorial on Using Yioop with Composer](#).

[Return to table of contents](#).

Localizing Yioop to a New Language

The Manage Locales activity can be used to configure Yioop for use with different languages and for different regions. If you decide to customize your Yioop installation by adding files to `WORK_DIRECTORY/app` as described in the [Building a Site using Yioop as a Framework](#) section, then the localization tools described in this section can also be used



name	tag	mode	Enabled	Percent	Actions
العربية	ar	rl-tb	true	100	Edit Delete
বাংলা	bn	lr-tb	true	100	Edit Delete
Deutsch	de	lr-tb	true	100	Edit Delete
English	en-US	lr-tb	true	100	Edit Delete
Español	es	lr-tb	true	100	Edit Delete
فارسی	fa	rl-tb	true	100	Edit Delete
Français	fr-FR	lr-tb	true	100	Edit Delete
עברית	he	rl-tb	true	100	Edit Delete
हिन्दी	hi	lr-tb	true	100	Edit Delete
Bahasa	id	lr-tb	true	100	Edit Delete
Italiano	it	lr-tb	true	100	Edit Delete
日本語	ja	lr-tb	true	100	Edit Delete
ಕನ್ನಡ	kn	lr-tb	true	100	Edit Delete
한국어	ko	lr-tb	true	100	Edit Delete
Nederlands	nl	lr-tb	true	100	Edit Delete
Polski	pl	lr-tb	true	100	Edit Delete
Português	pt	lr-tb	true	100	Edit Delete
Русский	ru	lr-tb	true	100	Edit Delete
తెలుగు	te	lr-tb	true	100	Edit Delete
ไทย	th	lr-tb	true	100	Edit Delete
Tagalog	tl	lr-tb	true	100	Edit Delete
Türkçe	tr	lr-tb	true	100	Edit Delete
Tiếng Việt	vi-VN	lr-tb	true	100	Edit Delete
中文	zh-CN	lr-tb	true	100	Edit Delete

Clicking the add icon above the table reveals the Add Locale Form:

Add Locale ?

Locale Name:

Locale Tag:

Writing Mode: Select Mode ?

Locale Enabled:

Save

This form allows you to create a new locale -- an object representing a language and a region. The first field on this form should be filled in with a name for the locale in the language of the locale. So for French you would put Français. The locale tag should be the IETF language tag. The **Writing Mode** element on the form is to specify how the language is written. There are four options: lr-tb -- from left-to-write from the top of the page to the bottom as in English, rl-tb from right-to-left from the top the page to the bottom as in Hebrew and Arabic, tb-rl from the top of the page to the bottom from right-to-left as in Classical Chinese, and finally, tb-lr from the top of the page to the bottom from left-to-right as in non-cyrillic Mongolian or American Sign Language. lr-tb and rl-tb support work better than the vertical language support. As of this writing, Internet Explorer and WebKit based



not to present the locale on the Settings page. This allows you to choose only the locales you want for your website without having to delete the locale data for other locales you don't want, but may want in the future as more translate strings become available. Beneath the Add Locale form is an alphabetical in the local-tah table listing some of the current locales. The Show Dropdown let's you control let's you control how many of these locales are displayed in one go. The Search link lets you bring up an advance search form to search for particular locales and also allows you to control the direction of the listing. The Locale List table first colume has a link with the name of the locale. Clicking on this link brings up a page where one can edit the strings for that locale. The next two columns of the Locale List table give the locale tag and writing direction of the locale, this is followed by the percent of strings translated. Clicking the Edit link in the column let's one edit the locale tag, and text direction of a locale. Finally, clicking the Delete link let's one delete a locale and all its strings.

To translate string ids for a locale click on its name link. This should display the following forms and table of string id and their transated values:

Edit Locale: English [X]

Show: Filter:

admin_controller_need_cookies	Cookies needed to login!!
admin_controller_account_not_active	Account not active. Maybe re
admin_controller_login_successful	Login Successful!!
admin_controller_login_failed	Username or Password Incon
admin_controller_login_to_config	Login to continue configurati
admin_controller_status_updates_stopped	Status updates have stopped
admin_controller_no_cookie_consent	Cookies need to be enabled t
admin_controller_account_access	Account
admin_controller_social	
admin_controller_crawl_settings	Crawls
admin_controller_system_settings	System Settings

In the above case, the link for English was clicked. The [X] link in the corner can be used to written to the previous form. The drop down controls whether to display all localizable strings or just those missing translations. The Filter field can be used to restrict the list of string id's and translations presented to just those containing what is in this field. Beneath this dropdown, the Edit Locale page mainly consists of a two column table: the right column being string ids, the left column containing what should be their translation into the given locale. If no translation exists yet, the field will be displayed in red. String ids are extracted by Yioop automatically from controller, view, helper, layout, and element class files which are either in the Yioop Installation itself or in the installation WORK_DIRECTORY/app folder. Yioop looks for tl() function calls to extract ids from these files, for example, on seeing tl('search_view_query_results') Yioop would extract the id search_view_query_results; on seeing tl('search_view_calculated', \$data['ELAPSED_TIME']) Yioop would extract the id, 'search_view_calculated'. In the second case, the translation is expected the translation to have a %s in it for the value of \$data['ELAPSED_TIME']. Note %s is used regardless of the the type, say int, float, string, etc., of \$data['ELAPSED_TIME']. tl() can handle additional arguments, whenever an additional argument is supplied an additional %s would be expected somewhere in the translation string. If you make a set of translations, be sure to submit the form associated with this table by scrolling to the bottom of the page and clicking the Submit link. This saves your translations; otherwise, your work will be lost if you navigate away from this page. One aid to translating is if you hover your mouse over a field that needs translation, then its translation in the default locale (usually English) is displayed. If you want to find where in



with the prefix `us_` (such as the names of activities) are stored in the database. So you cannot find these ids in the source code. The tooltip trick mentioned above does not work for database string ids.

Localizing Wiki Pages

When a user goes to a wiki page with a URL such as

```
YI00P_LOCATION/group/id_of_a_group/Some_Page_Name
```

or

```
YI00P_LOCATION/group/id_of_a_group/Some_Page_Name
```

or for the public group possibly with

```
YI00P_LOCATION/p/Some_Page_Name
```

the page that is displayed is in the locale that has been most recently set for the user. If no locale was set, then Yioop tries to determine the locale based on browser header info, and if this fails, falls back to the Default Locale set when Yioop was configure. When one edits a wiki page the locale that one is editing the page for is displayed under the page name such as en-US in the image below:



To edit the page for a different locale, choose the locale you want using the Settings page while logged in and then navigate to the wiki page you would like to edit (using the same name from the original language). Suppose you were editing the `Dental_Floss` page in en-US locale. To make the French page, you click Settings on the top bar of Yioop, go to your account settings, and choose French (fr-FR) as the language. Now one would navigate back to the wiki page you were on, the `Dental_Floss` page, which doesn't exist for French. You could click Edit now and make the French page at this location, but this would be sub-optimal as the French word for dental floss is `dentrifice`. So instead, on the fr-FR `Dental_Floss` edit page, you edit the page Settings to make this page a Page Alias for `Dentrifice`, and then create and edit the French `Dentrifice` article. If a user then starts on the English version of the page and switches locales to French they will end up on the `Dentrifice` page. You should also set up the page alias in the reverse direction as well, to handle when someone start on the French `Dentrifice` page and switches to the en-US `Dentrifice`.

Adding a stemmer, segmenter or supporting character n-gramming for your language

Depending on the language you are localizing to, it may make sense to write a stemmer for words that will be inserted into the index. A stemmer takes inflected or sometimes derived words and reduces them to their stem. For instance, jumps and jumping would be reduced to jump in English. As Yioop crawls it attempts to detect the language of a given web page it is processing. If a stemmer exists for this language it will call the Tokenizer class's `stem($word)` method on each word it extracts from the document before inserting information about it into the index. Similarly, if an end-user is entering a simple conjunctive search query and a stemmer exists for his language settings, then the query terms will be stemmed before being looked up in the index. Currently, Yioop comes with stemmers for Dutch, English, French, German, Hindi, Italian, Persian, Portuguese, Spanish, and Russian. The English stemmer uses the Porter Stemming Algorithm [[P1980](#)], the other stemmers are based on the algorithms presented at [snowballstem.org](#). Stemmers should be written as a static method located in the file `WORK_DIRECTORY/locale/en_US`

locale folder. For instance, one could modify the file `WORK_DIRECTORY/locale/
/in/resources/Tokenizer.php` to contain a class in `Tokenizer` with a static method `stem($word)` if one wanted to add a stemmer for Indonesian.

The class inside `Tokenizer.php` can also be used by Yioop to do word segmentation. This is the process of splitting a string of words without spaces in some language into its component words. Yioop comes with an example two segmenters for the zh-CN (Chinese) locale. The default, stochastic segmenter tries to predict the best possible splitting of the sentence based on statistics about the frequencies of terms over different possible splittings. It makes use of a `term_weights.txt.gz` to do this. The other segmenter works by starting at the end of the string and trying to greedily find the longest word that can be matched with the portion of the suffix of the string that has been processed yet (reverse maximal match). To do this it makes use of a word Bloom filter as part of how it detects if a string is a word or not. We describe how to make such filter using `TokenTool.php` in a moment.

In addition to supporting the ability to add stemmers and segmenters, Yioop also supports a default technique which can be used in lieu of a stemmer called character n-grams. When used this technique segments text into sequences of n characters which are then stored in Yioop as a term. For instance if n were 3 then the word "thunder" would be split into "thu", "hun", "und", "nde", and "der" and each of these would be associated with the document that contained the word thunder. N-grams are useful for languages like Chinese and Japanese in which words in the text are often not separated with spaces. It is also useful for languages like German which can have long compound words. The drawback of n-grams is that they tend to make the index larger. For Yioop built-in locales that do not have stemmer the file, the file `WORK_DIRECTORY/locale/LOCALE-TAG/resources/Tokenizer.php` has a line of the form

```
public static $char_gram_len = 5;
```

This number is the length of string to use in doing char-gramming. If you add a language to Yioop and want to use char gramming merely add a `tokenizer.php` to the corresponding locale folder with such a line in it.

Using `TokenTool.php` to improve search performance and relevance for your language

`configs/TokenTool.php` is used to create suggest word dictionaries, 'n' word gram (for word entities) filter files, knowledge wiki and seed site entries, segment filters, and named entity tag and part of speech tag files for the Yioop! search engine for a locale. It can also be used to localize Yioop text strings for a new language. To create dictionaries, filter, or tag files, the user puts a source file in Yioop's `WORK_DIRECTORY/prepare` folder. `TokenTool` will typically output the resulting file in the folder `LOCALE_DIR/locale_tag/resources` where `locale_tag` is the locale the file is being created for. Suggest word dictionaries are used to supply the content of the dropdown of search terms that appears as a user is entering a query in Yioop. To make a suggest dictionary one can use a command like:

```
php TokenTool.php dictionary filename locale endmarker
```

Here `filename` should be in the current folder or `PREP_DIR` and should consist of one word per line, `locale` is the locale this suggest (for example, en-US) file is being made for and where a file `suggest-trie.txt.gz` will be written, and `endmarker` is the end of word symbol to use in the trie. For example, `$` works pretty well.

`TokenTool.php` can also be used to make filter files. A filter file is used to detect when words in a language should be treated as a unit when extracting text during a crawl and at search time. For example, Bill Clinton is 2 word gram which should be treated as unit because it is a particular person. These filter files can also be used with a segmenter which might be used to split Chinese or Japanese text which does not have spaces into a sequence of Chinese and Japanese words (which may be made out of multiple characters). For a nonsegmenter filter, `TokenTool.php` is run from the command line as:

```
php TokenTool.php filter wiki_file lang locale n extract_type max_to_extract
```



segmenter case), locale is the IANA language tag of the locale to store the results for (it is different from lang, for example, en-US versus en for lang), n is the number of words in a row to consider, extract_type is where from Wikipedia source to extract:

```
0 = title's,  
1 = redirect's,  
2 = page count dump wikipedia data,  
3 = page count dump wiktionary data.
```

A knowledge wiki entry is a search wiki page which is displayed on a given query usually in a callout box. TokenTool.php can be used to create such entries based on the first paragraph of a Wikipedia page which matches the query. At the same time TokenTool.php is doing this it can also use the infoboxes on wiki pages to generate a initial list of potential seed sites for a web crawl. The syntax to create knowledge wiki seed sites is:

```
php TokenTool.php kwiki-seeds locale page_count_file wiki_locale_dump \  
num_entries num_seeds
```

Here locale is the IANA language tag of the locale to create knowledge wiki entries and seed sites for, page_count_file is a wiki page count dump file (or folder of such files), wiki_locale_dump is a dump of wikipedia pages and meta pages, num_entries says to get use the num_entries wiki pages according to what the page count file says were most frequently accessed and only make knowledge wiki entries for these, num_seeds says only use the infoboxes on the top num_seeds many pages as sources of seed site urls. Seed site urls will be written to the current crawl.ini file used to set the crawl parameters of the next crawl.

In some Asian languages, such as Chinese, Japanese, there are no spaces between words. In order to segment terms/words from a sentence, either a segmenter filter file or a stochastic segmenter file is used by Yioop. By default, Yioop tries to use stochastic segmentation to segment strings into phrases if a stochastic segmenter file for the locale exists. This is a slightly better approach. A segmenter filter file allows Yioop to split phrases according to reverse maximal match, which is a less accurate way to do segmentation. To create such files, TokenTool.php is run from the command line as:

```
php TokenTool.php stochastic-segmenter locale files_format max_files dataset_files...
```

or

```
php TokenTool.php segment-filter locale dictionary_file
```

respectively. Here locale is the IANA language tag of the locale to store the results for, files_format is the format of the files, currently supported format can be "default" or "CTB" (Chinese Tree Bank). The default format has all word segmented by space, CTB information can be found at:

```
https://www.cs.brandeis.edu/~clp/ctb/
```

max_files is the maximum number of found dataset files to process during training. If negative it means train on all files, if positive train at most that many files (useful if running out of memory). dataset_files... should be a list of text files or glob pattern to such text files with the format described above and dictionary_file is a text file or glob pattern to such text file with one word/line. An example is:

```
php TokenTool.php stochastic-segmenter zh-CN CTB -1 segmented/*
```

segment-filter outputs a file LOCALE_DIR/locale_tag/resources/segment.ftr and stochastic-segmenter outputs a file LOCALE_DIR/locale_tag/resources/term_weights.txt.gz TokenTool can be used to create name entity tag files. These can be used to find named entities in text passages for a language, and is used as part of the stochastic-segmenter process. The command to create such a file is:

```
php TokenTool.php entity-tagger locale tag_separator max_files dataset_files...
```

the arguments are the same as stochastic-segmenter except tag_separator. The input training files contain tagged white space separated terms. If the tag_separator was '-', then



recombine into this format. The file output by running the entity tagger command is
LOCALE_DIR/locale_tag/resources/nect_weights.txt.gz

TokenTool can also be used to part of speech tag files. These can be used for Yioop's question answering subsystem for a language. The command to create such a file is:

```
php TokenTool.php pos-tagger locale tag_separator max_files dataset_files...
```

The command line arguments are the same as the entity-tagger command. The file output by this command is: LOCALE_DIR/locale_tag/resources/pos_weights.txt.gz

Localizing Yioop's web app strings to a new language can be done manually via the Manage Locale activity within Yioop. Alternatively, TokenTool can be used to generate initial translations for these strings using the Yandex Translate API. To use this one needs to have in Yioop's src/configs/LocalConfig.php file a line like:

nsdefine("TRANSLATE_API_KEY", "the value of your Yandex Translate API key"); Then to translate Yioop web app strings to a locale one can do:

```
php TokenTool.php translate-locale locale
```

or

```
php TokenTool.php translate-locale locale with_wiki_pages
```

Here locale is the IANA language tag of the locale to translate. If locale is set to "all", then translations will be done for all locales in the Yioop system (excluding English, which is where the translations are coming from). with_wiki_pages controls how public and help wiki pages are translated. By default and if this is <=0, public and help wiki pages are not translated, if it is 1, they are translated to the locale if the locale does not already have a translation. If it is >1 then it is force translated to locale. After this program finishes, you should do a sanity check on the translations using Manage Locales. Remember if you hover above a translated item in the Edit string interface it displays as a tooltip the original English text.

Obtaining data sets for TokenTool.php

Many word lists with frequencies are obtainable on the web for free with Creative Commons licenses. A good Many word lists are obtainable on the web for free with Creative Commons licenses. A good starting point is:

```
http://en.wiktionary.org/wiki/Wiktionary:Frequency\_lists
```

A little script-fu can generally take such a list and put it into the format of one word/term per line which is needed by TokenTool.php

For filter file, page count dumps can be found at

```
https://dumps.wikimedia.org/other/pagecounts-ez/merged/
```

These probably give the best n-gram or all gram results, usually in a matter of minutes; nevertheless, this tool does support trying to extract similar data from Wikipedia dumps. This can take hours.

For Wikipedia dumps, one can go to <https://dumps.wikimedia.org/enwiki/> and obtain a dump of the English Wikipedia (similar for other languages). This page lists all the dumps according to date they were taken. Choose any suitable date or the latest. A link with a label such as 20120104/, represents a dump taken on 01/04/2012. Click this link to go in turn to a page which has many links based on type of content you are looking for. For this tool you are interested in files under "Recombine all pages, current versions only".

Beneath this we might find a link with a name like:

```
enwiki-20120104-pages-meta-current.xml.bz2
```

which is a file that could be processed by this tool.

A Creative Commons licensed file which can be manipulated into a dictionary file suitable for Chinese segmentation can be found at:



queries are mainly restricted to fixing typos in the original query. In order to calculate these spelling corrections, Yioop takes the query and for each query term computes each possible single character change to that term. For each of these it looks up in the given locale's suggest_trie.txt.gz a frequency count of that variant, if it exists. If the best suggestion is some multiple better than the frequency count of the original query then Yioop suggests this alternative query. In order for this to work, Yioop needs to know what constitutes a single character in the original query. The file locale.js in the WORK_DIRECTORY/locale/LOCALE_TAG/resources folder can be used to specify this for the locale given by LOCALE_TAG. To do this, all you need to do is specify a Javascript variable alpha. For example, for French (fr-FR) this looks like:

```
var alpha = "aâàbccçdeééfghiiijklmnoôpqrstuûvwxyz";
```

The letters do not have to be in any alphabetical order, but should be comprehensive of the non-punctuation symbols of the language in question.

Another thing locale.js can be used for is to given mappings between roman letters and other scripts for use in the Yioop's autosuggest dropdown that appears as you type a query. As you type, scripts/suggest.js function onTypeTerm is called. This in turn will cause a particular locale's locale.js function transliterate(query) if it exists. This function should return a string with the result of the transliteration. An example of doing this is given for the Telugu locale in Yioop.

Part of Speech Tagging

Yioop may do part of speech tagging on both queries and text during summarization as part of its question answering system. I.e., in order to be able to answer a what, when, where, etc question, Yioop needs to be able to the know the known noun, verb, etc of the sentence. For the en-US locale, this is done by calling a method:

tagTokenizePartOfSpeech(\$text) in WORK_DIRECTORY/locale/en_US/resources/Tokenizer.php. For en-US, a simple Brill tagger (see Ranking document for more info) is implemented to do this. After this method is called the terms in \$text should have a suffix ~part-of-speech where ~part-of-speech where part-of-speech is one of NN for noun, VB for verb, AJ for adjective, AV for adverb, or some other value (which would be ignored by Yioop). For example, the noun dog might become dog~NN after tagging. To localize to another language this method in the corresponding Tokenizer.php file would need to be implemented. The process of going from tagged parts of speech into an answer for a question is down using a triplet mechanism where by the query is converted into a a triplet, for example, [qqq is prime minister]. A search is done on this triplet. Then documents which had this triplet are returned. For those documents, it is assumed when the triplet was indexed, a value to fill in for qqq was also determined. For example, for some document qqq may have been indexed with substitution Theresa May. Locales that support question answering need to implement in their Tokenizer.php, the method extractTripletsPhrases. This currently has been done for en-US, hi (Hindi), zh-CN (Chinese).

Using Stop Words for Locale Detection and Summarization

While crawling, Yioop makes use of a summarizer to extract the important portions of the web page both for indexing and for search result snippet purposes. There are three summarizers that come with Yioop a Basic summarizer, which uses an ad hoc approach to finding the most important parts of the document; a Graph-based summarizer, which convert the sentences into vertices, connecting them with weighted edges, and tries to a page-rank like computation, and a centroid summarizer which tries to compute an "average sentence" for the document and uses this to pick representative sentences based on nearness to this average. The summarizer that is used can be set under the Crawl Time tab of [Page Options](#) . This centroid summarizer works better if certain common words (stop words) from the documents language are removed. When using the centroid summarizer, Yioop check to see if tokenizer.php for the current locale contains a method stopwordsRemover(\$page). If it does it calls it, this method takes a string of words are returns a string with all the stop words removed. This method exists for en-US, but, if



Modifying Yioop Code

One advantage of an open-source project is that you have complete access to the source code. Thus, Yioop can be modified to fit in with your existing project. You can also freely add new features onto Yioop. In this section, we look a little bit at some of the common ways you might try to modify Yioop as well as ways to examine the output of a crawl in a more technical manner. If you decide to modify the source code it is recommended that you look at the [Summary of Files and Folders](#) above again, as well as look at the [online Yioop code documentation](#).

Handling new File Types

One relatively easy enhancement to Yioop is to enhance the way it processes an existing file type or to get it to process new file types. Yioop was written from scratch without dependencies on existing projects. So the PHP processors for Microsoft file formats and for PDF are only approximate. These processors can be found in `src/library/processors`. To write your own processor, you should extend either the `TextProcessor` or `ImageProcessor` class. You then need to write in your subclass a static method `process($page, $url)`. Here `$page` is a string representation of a downloaded document of the file type you are going to handle and `$url` is the canonical url from which this page is downloaded. This method should return an array of the format:

```
$summary['TITLE'] = a title for the document
$summary['DESCRIPTION'] = a text summary extracted from the document
$summary['LINKS'] = an array of links (canonical not relative) extracted
                    from the document.
```

A good reference implementation of a `TextProcessor` subclass can be found in `HtmlProcessor.php`. If you are trying to support a new file type, then to get Yioop to use your processor you need to add lines to your classes constructor specifying the `self::$indexed_file_types` and `self::$mime_processor` static fields. As an example, in the `RssProcessor` class; constructor these lines are:

```
self::$indexed_file_types[] = "rss";
self::$mime_processor["application/rss+xml"] = "RssProcessor";
self::$mime_processor["application/atom+xml"] = "RssProcessor";
```

`self::$indexed_file_types` says that there exists a processor for a given file extension, `self::$mime_processor` says what processor should be used to handle a given mime type. Both these fields are static and inherited from the base class. They are used to get effectively global array of all handled file types and which processors should be used for which extensions. If you create a new processor, it is cool, only relies on code you wrote, and you want to contribute it back to the Yioop, please feel free to e-mail it to chris@pollett.org.

Writing a MediaJob

A `MediaJob` is an activity to be performed by the `MediaUpdater` process on a periodic basis. Currently, three `MediaJob`'s have already been created in the `library/media_jobs` folder: `NewsUpdateJob`, used to update the news feeds which show in the News subsearch; `BulkEmailJob`, used to send out emails to thread or group participants when an update has occurred to that thread or group; and `VideoConvertJob`, which is used to convert uploaded videos to a web streaming friendly mp4 format. One can write one's own `MediaJob`'s for any activity that one would like Yioop to do on a periodic basis. One simply creates a subclass of `MediaJob` and places it in the `WORK_DIRECTORY/app/library/media_jobs` folder. The code for the three `MediaJobs` above serve as a useful starting point for writing such jobs. Below though is a summary of the principal methods that a `MediaJob` needs to have in order for it to be run by the `MediaUpdater`:

init()

called after the class' constructor and allows a user entry point to perform initializations for the job.

checkPrerequisites()



is called only on the name server and only called when media updater is run in non-distributed mode. You can put the nondistributed version of your job in this method.

prepareTasks()

is called on the name server's media updater only, but is intended to be used in the distributed MediaUpdater setting. It is supposed to get the data needed by the job ready before it is sent to a client machine's MediaUpdater.

getTasks()

is called by the name server's web app when a client machine's MediaUpdater makes an HTTP request for data for the MediaJob. It takes data output by prepareTasks() and sends the client its portion of this data (we imagine that a data set might be split between several clients).

doTasks()

is called after the client MediaUpdater has received the getTasks() data. It then processes this data.

putTasks()

is called by the name server's web app when a client makes an HTTP request to send processed information back to the web server.

finishTasks()

is called the name server's MediaUpdater only. It performs any final computations needed on data after it has been sent back to name server.

Writing an Indexing Plugin

An indexing plugin provides a way that an advanced end-user can extend the indexing capabilities of Yioop. Bundled with Yioop are three example indexing plugins. These are found in the library/indexing_plugins folder. We will discuss the code for the recipe and word filter plugin here. The code for the address plugin, used to extract snail mail address from web pages follows the same kind of structure. If you decide to write your own plugin or want to install a third-party plugin you can put it in the folder: WORK_DIRECTORY/app/library/indexing_plugins. The recipe indexing plugin can serve as a guide for writing your own plugin if you don't need your plugin to have a configure screen. The recipe plugin is used to detect food recipes which occur on pages during a crawl. It creates "micro-documents" associated with found recipes. These are stored in the index during the crawl under the meta-word "recipe:all". After the crawl is over, the recipe plugin's postProcessing method is called. It looks up all the documents associated with the word "recipe:all". It extracts ingredients from these and does clustering of recipes based on ingredient. It finally injects new meta-words of the form "ingredient:some_food_ingredient", which can be used to retrieve recipes most closely associated with a given ingredient. As it is written, recipe plugin assumes that all the recipes can be read into memory in one go, but one could easily imagine reading through the list of recipes in batches of the amount that could fit in memory in one go.

The recipe plugin illustrates the kinds of things that can be written using indexing plugins. To make your own plugin, you would need to write a subclass of the class IndexingPlugin with a file name of the form MypluginnamePlugin.php. Then you would need to put this file in the folder WORK_DIRECTORY/app/library/indexing_plugins. RecipePlugin subclasses IndexingPlugin and implements the following four methods: pageProcessing(\$page, \$url), postProcessing(\$index_name), getProcessors(), getAdditionalMetaWords() so they don't have their return NULL default behavior. We explain what each of these is for in a moment. During a web crawl, after a fetcher has downloaded a batch of web pages, it uses a page's mimetype to determine a page processor class to extract summary data from that page. The page processors that Yioop implements can be found in the folder lib/processors. They have file names of the form someprocessorname_processor.php. As a crawl proceeds, your plugin will typically be called to do further processing of a page only in addition to some of these processors. The static method getProcessors() should return an array of the form ["someprocessorname1", "someprocessorname2", ...], listing the processors that your



pageProcessing(\$page, \$url) of each indexing_plugin associated with the given processor. A pageProcessing(\$page, \$url) method is expected to return an array of subdoc arrays found on the given page. Each subdoc array should have a CrawlConstants::TITLE and a CrawlConstants::DESCRIPTION. The handle method of a processor will add to each subdoc the fields: CrawlConstants::LANG, CrawlConstants::LINKS, CrawlConstants::PAGE, CrawlConstants::SUBDOCTYPE. The SUBDOCTYPE is the name of the plugin. The resulting "micro-document" is inserted by Yioop into the index under the word nameofplugin:all . After the crawl is over, Yioop will call the postProcessing(\$index_name) method of each indexing plugin that was in use. Here \$index_name is the timestamp of the crawl. Your plugin can do whatever post processing it wants in this method. For example, the recipe plugin does searches of the index and uses the results of these searches to inject new meta-words into the index. In order for Yioop to be aware of the meta-words you are adding, you need to implement the method getAdditionalMetaWords(). Also, the web snippet you might want in the search results for things like recipes might be longer or shorter than a typical result snippet. The getAdditionalMetaWords() method also tells Yioop this information. For example, for the recipe plugin, getAdditionalMetaWords() returns the associative array:

```
["recipe:" => HtmlProcessor::MAX_DESCRIPTION_LEN,  
 "ingredient:" => HtmlProcessor::MAX_DESCRIPTION_LEN]
```

The WordFilterPlugin illustrates how one can write an indexing plugin with a configure screen. It overrides the base class' pageSummaryProcessing(&\$summary) and getProcessors() methods as well as implements the methods saveConfiguration(\$configuration), loadConfiguration(), setConfiguration(\$configuration), configureHandler(&\$data), and configureView(&\$data). The purpose of getProcessors() was already mentioned under recipe plugin description above. pageSummaryProcessing(&\$summary) is called by a page processor after a page has been processed and a summary generated. WordFilterPlugin uses this callback to check if the title or the description in this summary have any of the words the filter is filtering for and if so takes the appropriate action. loadConfiguration, saveConfiguration(\$configuration), and setConfiguration are three methods to handle persistence for any plugin data that the user can change. The first two operate on the name server, the last might operate on a queue server or a fetcher. loadConfiguration is be called by configureHandler(&\$data) to read in any current configuration, unserialize it and modify it according to any data sent by the user. saveConfiguration(\$configuration) would then be called by configureHandler(&\$data) to serialize and write any \$configuration data that needs to be stored by the plugin. For WordFilterPlugin, a list of filter terms and actions are what is saved by saveConfiguration(\$configuration) and loaded by loadConfiguration. When a crawl is started or when a fetcher contacts the name server, plugin configuration data is sent by the name server. The method setConfiguration(\$configuration) is used to initialize the local copy of a fetcher's or queue server's process with the configuration settings from the name server. For WordFilterPlugin, the filter terms and actions are stored in a field variable by this function. As has already been hinted at by the configuration discussion above, configureHandler(&\$data) plays the role of a controller for an index plugin. It is in fact called by the AdminController activity pageOptions if the configure link for a plugin has been clicked. In addition, to managing the load and save configuration process, it also sets up any data needed by configureView(&\$data). For WordFilterPlugin, this involves setting a variable \$data["filter_words"] so that configureView(&\$data) has access to a list of filter words and actions to draw. Finally, the last method of the WordFilterPlugin we describe, configureView(&\$data), outputs using \$data the HTML that will be seen in the configure screen. This HTML will appear in a div tag on the final page. It is initially styled so that it is not displayed. Clicking on the configure link will cause the div tag data to be displayed in a light box in the center of the screen. For WordFilterPlugin, this methods draws a title and



can page option and page option will know in turn to let program's configuration handle methods to get a chance to handle this data.

[Return to table of contents](#) .

Yioop Command-line Tools

In addition to [TokenTool.php](#) which we describe in the section on localization, and to [ExportPublicHelpDb.php](#) and [GroupWikiTool.php](#) which we describe in the section on the Yioop folder structure, Yioop comes with several useful command-line tools and utilities. We next describe these in roughly their order of likely utility:

- [src/configs/ConfigureTool.php](#) : Used to configure Yioop from the command-line
- [src/executables/ArcTool.php](#) : Used to examine the contents of WebArchiveBundle's, IndexArchiveBundles's, and DoubleIndexBundles's
- [src/executables/QueryTool.php](#) : Used to query an index from the command-line
- [src/executables/CodeTool.php](#) : Used to help code Yioop and to help make clean patches for Yioop.
- [src/executables/ClassifierTool.php](#) : Used to make Yioop a Yioop classifier from the command line rather than using the GUI interface.

Configuring Yioop from the Command-line

In a multiple queue server and fetcher setting, one might have web access only to the name server machine -- all the other machines might be on virtual private servers to which one has only command-line access. Hence, it is useful to be able to set up a work directory and configure Yioop through the command-line. To do this one can use the script `configs/ConfigureTool.php`. One can run it from the command-line within the configs folder, with a line like:

```
php ConfigureTool.php
```

When launched, this program will display a menu like:

```
Yioop CONFIGURATION TOOL
+++++

Checking Yioop configuration...
=====
Check Passed.
Using configs/LocalConfig.php so changing work directory above may not work.
=====

Available Options:
=====
(1) Create/Set Work Directory
(2) Change root password
(3) Set Default Locale
(4) Debug Display Set-up
(5) Search Access Set-up
(6) Search Page Elements and Links
(7) Name Server Set-up
(8) Crawl Robot Set-up
(9) Exit program

Please choose an option:
```

Except for the Change root password option, these correspond to the different fieldsets on the Configure activity. The command-line forms one gets from selecting one of these choices let one set the same values as were described earlier in the [Installation](#) section. The change root password option lets one set the account password for root. I.e., the main admin user. On a non-nameserver machine, it is probably simpler to go with a sqlite database, rather than hit on a global mysql database from each machine. Such a barebones local database set-up would typically only have one user, root. Another thing to consider when configuring a collection of Yioop machines in such a setting, is, by default, under Search Access Set-up, subsearch is unchecked. This means the RSS feeds won't be downloaded hourly on such machines. If one unchecks this, they can be.

Examining the contents of WebArchiveBundle's and IndexArchiveBundles's



php ArcTool.php count bundle_name
php ArcTool.php count double_index_name which_bundle

or

php ArcTool.php count bundle_name save
php ArcTool.php count double_index_name which_bundle save
returns the counts of docs and links for each shard in bundle as well as an overall total.
The second command saves the just computed count into the index description (can be used to fix the index count if it gets screwed up).

php ArcTool.php dict bundle_name word
php ArcTool.php dict double_index_name which_bundle word
php ArcTool.php dict bundle_name word start_gen num_gen
php ArcTool.php dict double_index_name which_bundle word start_gen num_gen
returns index dictionary records for word stored in index archive bundle or double index bundle. In the later case you should provide which bundle you want dictionary info for.
This command also supports startand number of generation parameters.

php ArcTool.php info bundle_name
return info about documents stored in archive.

php ArcTool.php inject timestamp file
injects the urls in file as a schedule into crawl of given timestamp. This can be used to make a closed index unclosed and to allow for continued crawling.

php ArcTool.php list
returns a list of all the archives in the Yioop! crawl directory, including non-Yioop! archives in the /archives sub-folder.

php ArcTool.php mergetiers bundle_name max_tier
merges tiers of word dictionary into one tier up to max_tier

php ArcTool.php posting bundle_name generation offset
php ArcTool.php posting double_index_name which_bundle generation offset

or

php ArcTool.php posting bundle_name generation offset num
php ArcTool.php posting double_index_name which_bundle generation offset num
returns info about the posting (num many postings) in bundle_name at the given generation and offset

php ArcTool.php rebuild bundle_name
php ArcTool.php rebuild double_index_name which_bundle
Re-extracts words from summaries files in bundle_name into index shards then builds a new dictionary

php ArcTool.php reindex bundle_name
php ArcTool.php reindex bundle_name start_shard
Reindex the word dictionary in bundle_name using existing index shards. In the second version it is assumed that the index has been correctly reindexed already to *start_shard - 1* , and the command continues the reindexing from *start_shard*

php ArcTool.php shard bundle_name generation
php ArcTool.php shard double_index_name which_bundle generation
Prints information about the number of words and frequencies of words within the generation'th index shard in the bundle or double index bundle (in which case need to say either 0 or 1 bundle)

php ArcTool.php show bundle_name start num
php ArcTool.php show double_index_name which_bundle start num
outputs items start through num from bundle_name or name of non-Yioop archive crawl folder

The bundle name can be a full path name, a relative path from the current directory, or it

image files where it is indicated below as being an archive file. They are not all from the same session:

```
|chris-polletts-macbook-pro:executables:108>php ArcTool.php list
Found Yioop Archives:
=====
0-Archive1334468745
0-Archive1336527560
IndexData1334468745
IndexData1336527560

Found Non-Yioop Archives:
=====
english-wikipedia2012

...

|chris-polletts-macbook-pro:executables:158>php ArcTool.php info \
/Applications/XAMPP/xamppfiles/htdocs/crawls/cache/IndexData1293767731

Bundle Name: IndexData1293767731
Bundle Type: IndexArchiveBundle
Description: test
Number of generations: 1
Number of stored links and documents: 267260
Number of stored documents: 16491
Crawl order was: Page Importance
Seed sites:
  http://www.ucanbuyart.com/
  http://www.ucanbuyart.com/fine_art_galleries.html
  http://www.ucanbuyart.com/indexucba.html
Sites allowed to crawl:
  domain:ucanbuyart.com
  domain:ucanbuyart.net
Sites not allowed to be crawled:
  domain:arxiv.org
  domain:ask.com
Meta Words:
  http://www.ucanbuyart.com/(.+)/(.+)/(.+)/(.+)/

|chris-polletts-macbook-pro:executables:202>php ArcTool.php show \
/Applications/XAMPP/xamppfiles/htdocs/crawls/cache/Archive1293767731 0 3

BEGIN ITEM, LENGTH:21098
[URL]
http://www.ucanbuyart.com/robots.txt
[HTTP RESPONSE CODE]
404
[MIMETYPE]
text/html
[CHARACTER ENCODING]
ASCII
[PAGE DATA]
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>
  <base href="http://www.ucanbuyart.com/" />
</pre>
...

|chris-polletts-macbook-pro:bin:117>php arc_tool.php reindex IndexData1317414152

Shard 0
[Sat, 01 Oct 2011 11:05:17 -0700] Adding shard data to dictionary files...
[Sat, 01 Oct 2011 11:05:28 -0700] Merging tiers of dictionary

Final Merge Tiers

Reindex complete!!
```

The mergetiers command is like a partial reindex. It assumes all the shard words have been added to the dictionary, but that the dictionary still has more than one tier (tiers are the result of incremental log-merges which are made during the crawling process). The mergetiers command merges these tiers into one large tier which is then usable by Yioop for query processing.

Querying an Index from the command-line

The command-line script `src/executables/QueryTool.php` can be used to query indices in



arguments for the query tool are:

```
php QueryTool.php query num_results start_num lang_tag
```

The default num_results is 10, start_num is 0, and lang_tag is en-US. The following shows how one could do a query on "Chris Pollett":

```
|chris-polletts-macbook-pro:executables:141>php QueryTool.php "Chris Pollett"
```

```
=====
```

```
TITLE: ECCC - Pointers to  
URL: http://eccc.hpi-web.de/static/pointers/personal_www_home_pages_of_complexity_theorists/  
IPs: 141.89.225.3  
DESCRIPTION: Homepage of the Electronic Colloquium on Computational Complexity located  
at the Hasso Plattner Institute of Potsdam, Germany Personal WWW pages of  
complexity people 2011 2010 2009 2011...1994 POINTE  
Rank: 3.9551158411  
Relevance: 0.49244377769  
Proximity: 1  
Score: 4.14  
=====
```

```
=====
```

```
TITLE: ECCC - Pointers to  
URL: http://www.eccc.uni-trier.de/static/pointers/personal_www_home_pages_of_complexity_theorists/  
IPs: 141.89.225.3  
DESCRIPTION: Homepage of the Electronic Colloquium on Computational Complexity located  
at the Hasso Plattner Institute of Potsdam, Germany Personal WWW pages of  
complexity people 2011 2010 2009 2011...1994 POINTE  
Rank: 3.886318974  
Relevance: 0.397622570289  
Proximity: 1  
Score: 4.03  
=====
```

```
.....
```

The index the results are returned from is the default index; however, all of the Yioop meta words should work so you can do queries like "my_query i:timestamp_of_index_want". Query results depend on the kind of language stemmer/char-gramming being used, so French results might be better if one specifies fr-FR then if one relies on the default en-US. QueryTool can also be used to explain what index iterators Yioop will use to execute a query. To see this information one can type a command like:

```
php QueryTool.php query plan
```

or

```
php QueryTool.php query explain
```

for example,

```
php QueryTool.php 'chris pollett' explain
```

A Tool for Coding and Making Patches for Yioop

src/executables/CodeTool.php can perform several useful tasks to help developers program for the Yioop environment. Below is a brief summary of its functionality:

php CodeTool.php clean path

Replaces all tabs with four spaces and trims all whitespace off ends of lines in the folder or file path.

php CodeTool.php copyright path

Adjusts all lines in the files in the folder at path (or if path is a file just that) of the form 2009 - \d\d\d\d to the form 2009 - this_year where this_year is the current year.

php CodeTool.php longlines path

Prints out all lines in files in the folder or file path which are longer than 80 characters.

php CodeTool.php replace path pattern replace_string

or

php CodeTool.php replace path pattern replace_string effect



Prints each line matching the regular expression pattern followed by the result of replacing pattern with replace_string in the folder or file path. Then it asks if you want to update the line. Lines you choose for updating will be modified in the files.

php CodeTool.php replace path pattern replace_string change

Each line matching the regular expression pattern is update by replacing pattern with replace_string in the folder or file path. This format doe not echo anything, it does a global replace without interaction.

php CodeTool.php search path pattern

Prints all lines matching the regular expression pattern in the folder or file path.

A Command-line Tool for making Yioop Classifiers

src/executables/ClassifierTool.php is used to automate the building and testing of classifiers, providing an alternative to the web interface when a labeled training set is available.

ClassifierTool.php takes an activity to perform, the name of a dataset to use, and a label for the constructed classifier. The activity is the name of one of the 'run*' functions implemented by this class, without the common 'run' prefix (e.g., 'TrainAndTest' for the method runTrainAndTest). The dataset is specified as the common prefix of two indexes that have the suffixes "Pos" and "Neg", respectively. So if the prefix were "DATASET", then this tool would look for the two existing indexes "DATASET Pos" and "DATASET Neg" from which to draw positive and negative examples. Each document in these indexes should be a positive or negative example of the target class, according to whether it's in the "Pos" or "Neg" index. Finally, the label is just the label to be used for the constructed classifier. Beyond these options (set with the -a, -d, and -l flags), a number of other options may be set to alter parameters used by an activity or a classifier. These options are set using the -S, -I, -F, and -B flags, which correspond to string, integer, float, and boolean parameters respectively. These flags may be used repeatedly, and each expects an argument of the form NAME=VALUE, where NAME is the name of a parameter, and VALUE is a value parsed according to the flag. The NAME should match one of the keys of the options member of this class, where a period ('.') may be used to specify nesting. For example:

```
-I debug=1          # set the debug level to 1
-B cls.use_nb=0     # tell the classifier to use Naive Bayes
```

To build and evaluate a classifier for the label 'spam', trained using the two indexes "DATASET Neg" and "DATASET Pos", and a maximum of the top 25 most informative features:

```
php ClassifierTool.php -a TrainAndTest -d 'DATASET' -l 'spam' -I cls.chi2.max=25
```

The above assume we are in the folder of ClassifierTool.php.

References

[APC2003]

Serge Abiteboul and Mihai Preda and Gregory Cobena. [Adaptive on-line page importance computation](#). In: Proceedings of the 12th international conference on World Wide Web. pp.280-290. 2003.

[B1970]

Bloom, Burton H. [Space/time trade-offs in hash coding with allowable errors](#). Communications of the ACM Volume 13 Issue 7. pp. 422–426. 1970.

[BSV2004]

Paolo Boldi and Massimo Santini and Sebastiano Vigna. [Do Your Worst to Make the Best: Paradoxical Effects in PageRank Incremental Computations](#). Algorithms and Models for the Web-Graph. pp. 168–180. 2004.

[BP1998]

Brin, S. and Page, L. [The Anatomy of a Large-Scale Hypertextual Web Search Engine](#). In: Seventh International World-Wide Web Conference (WWW 1998). April 14-18,



- [and Evaluating Search Engines. MIT Press. 2010.](#)
- [DG2004]**
Jeffrey Dean and Sanjay Ghemawat. [MapReduce: Simplified Data Processing on Large Clusters](#). OSDI'04: Sixth Symposium on Operating System Design and Implementation. 2004
- [GGL2003]**
Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. [The Google File System](#). 19th ACM Symposium on Operating Systems Principles. 2003.
- [GLM2007]**
A. Genkin, D. Lewis, and D. Madigan. [Large-scale bayesian logistic regression for text categorization](#). Technometrics. Volume 49. Issue 3. pp. 291--304, 2007.
- [H2002]**
T. Haveliwala. [Topic-Sensitive PageRank](#). Proceedings of the Eleventh International World Wide Web Conference (Honolulu, Hawaii). 2002.
- [KSV2010]**
Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. [A Model of Computation for MapReduce](#). Proceedings of the ACM Symposium on Discrete Algorithms. 2010. pp. 938-948.
- [KC2004]**
Rohit Khare and Doug Cutting. [Nutch: A flexible and scalable open-source web search engine](#). CommerceNet Labs Technical Report 04. 2004.
- [LDH2010]**
Jimmy Lin and Chris Dyer. [Data-Intensive Text Processing with MapReduce](#). Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers. 2010.
- [LM2006]**
Amy N. Langville and Carl D. Meyer. [Google's PageRank and Beyond](#). Princeton University Press. 2006.
- [MRS2008]**
C. D. Manning, P. Raghavan and H. Schütze. [Introduction to Information Retrieval](#). Cambridge University Press. 2008.
- [MKSR2004]**
G. Mohr, M. Kimpton, M. Stack, and I.Ranitovic. [Introduction to Heritrix, an archival quality web crawler](#). 4th International Web Archiving Workshop. 2004.
- [PTSHVC2011]**
Manish Patil, Sharma V. Thankachan, Rahul Shah, Wing-Kai Hon, Jeffrey Scott Vitter, Sabrina Chandrasekaran. [Inverted indexes for phrases and strings](#). Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp 555--564. 2011.
- [P1997a]**
J. Peek. [Summary of the talk: The AltaVista Web Search Engine](#) by Louis Monier. USENIX Annual Technical Conference Anaheim. California. ;login: Volume 22. Number 2. April 1997.
- [P1997b]**
J. Peek. [Summary of the talk: The Inktomi Search Engine by Louis Monier](#). USENIX Annual Technical Conference. Anaheim, California. ;login: Volume 22. Number 2. April 1997.
- [P1994]**
B. Pinkerton. [Finding what people want: Experiences with the WebCrawler](#). In Proceedings of the First World Wide Web Conference, Geneva, Switzerland. 1994.
- [P1980]**
M.F. Porter. [An algorithm for suffix stripping](#). Program. Volume 14 Issue 3. 1980. pp 130-137. On the same website, there are [stemmers for many other languages](#).
- [PDGQ2006]**
Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan. [Interpreting the Data:](#)



[ZCTSR2004]

Tom White. [Hadoop: The Definitive Guide](#). O'Reilly. 2009.

[ZCTSR2004]

Hugo Zaragoza, Nick Craswell, Michael Taylor, Suchi Saria, and Stephen Robertson. [Microsoft Cambridge at TREC-13: Web and HARD tracks](#). In Proceedings of 3th Annual Text Retrieval Conference. 2004.

[Return to table of contents](#) .

(c) 2022 Seekquarry, LLC - [Open Source Search Engine Software](#). [About Seekquarry](#) .