Part B: Recipe Search Engine Interface

Repository

https://github.com/rmace001/RecipeSearchEngine

Team Members

Rogelio Macedo - rmace001@ucr.edu
Jerry Tan - jtan027@ucr.edu
Erin Wong DeForest - ewong012@ucr.edu
Shiyao Feng - sfeng023@ucr.edu

Collaboration Details

Map Reduce/ Data Scripts

Jerry Tan
Entirely responsible for developing multiple scripts to change the formatting of overall recipe data on elastic search index, as well as expressing the hadoop index into key value pairs in a csv file. Furthermore, the scripts handled all data-cleaning and preprocessing, as well as schema conversions in effort to allow for sorting of recipe results by active time and total times.

Web Interface Backend/Frontend

Rogelio Macedo & Shiyao Feng

Elastic Enterprise App Search API

Set up a deployment on the Elastic Cloud Service, along with an Enterprise Search API engine

ReactJS Web Application

Provides an interface for querying the Elastic App Search API.

Hadoop MapReduce Index Insertion in Cassandra

Instantiation and table creation with Apache Cassandra on an Ubuntu 18.04.5 virtual machine.

Express, NodeJS Web Application

Provides an interface for querying the Apache Cassandra, along with ranking and displaying of documents.

Hadoop MapReduce Index Generation

Erin Wong DeForest
Used Hadoop on a single cluster and wrote MapReduce program to create the inverted index of recipe terms from the json data. Reformatted the data for input into Cassandra database for use in querying.

Overview of System

Architecture

We use Hadoop and MapReduce to create an inverted index of recipe terms where the document IDs are recipe links. We built our search engine frontend using Elastic Enterprise App Search with React as an interface. In the backend, we use Cassandra DB to store our inverted index and recipe data to return and display results upon querying. Since we are creating a search engine for culinary purposes, we have chosen to return queries based on the intersection of search terms. We made this choice because people searching for recipes often do so based on specific ingredients they have available and wish to cook with, for example “rice coconut milk”.

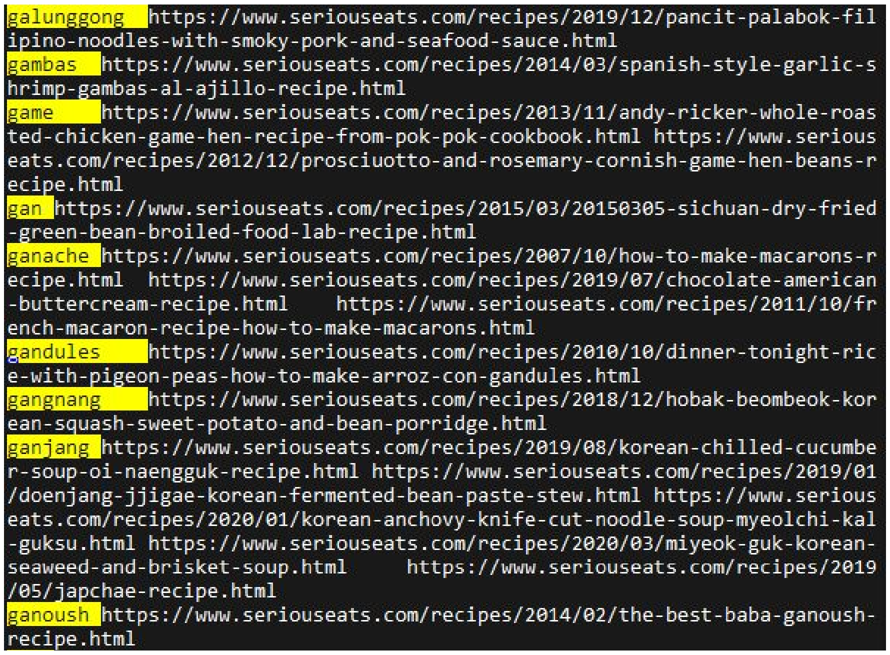
Hadoop MapReduce Indexing

We ran Hadoop on a single cluster to process our dataset of recipes. The program begins by editing the input file(s) to be one line per recipe and removing active and total time and JSON fields and punctuation as well as inserting tabs as delimiters between each field. Afterwards, MapReduce is used to create an inverted index where recipe links are used as document IDs because they are unique. All terms from text fields are used as keys; again, this only excludes active and total time.

While we removed punctuation, we chose not to perform stemming because many recipes contained foreign words which may be unintentionally altered by a stemmer. We handled foreign characters by using UTF-8.

The Mapper receives one recipe at a time, separating the link from the recipe and tokenizing all other text. The tokens are then emitted along with the recipe link. The Reducer compiles the inverted index for each term by appending recipe links.

The runtime of the Hadoop index construction was consistently about 25 seconds. There is no comparison of runtime to the Lucene Index because our ElasticSearch index was created automatically by uploading to the Elastic Enterprise App Search Server.

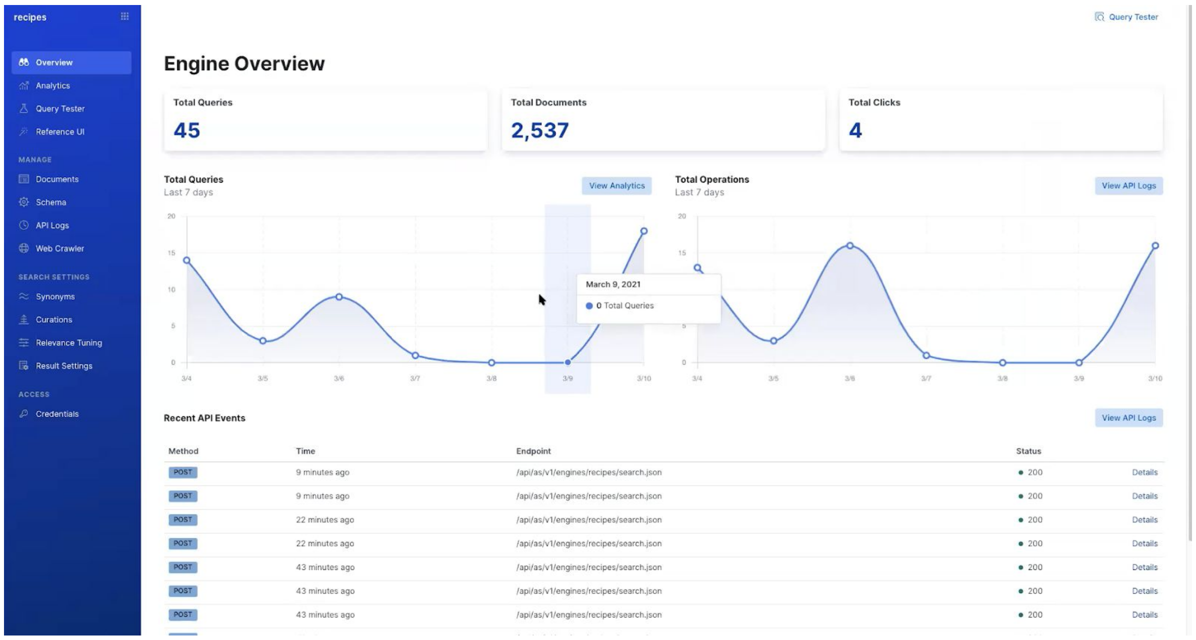


```
galunggong  https://www.seriouseats.com/recipes/2019/12/pancit-palabok-fil
ipino-noodles-with-smoky-pork-and-seafood-sauce.html
gambas  https://www.seriouseats.com/recipes/2014/03/spanish-style-garlic-s
hrimp-gambas-al-ajillo-recipe.html
game    https://www.seriouseats.com/recipes/2013/11/andy-ricker-whole-roas
ted-chicken-game-hen-recipe-from-pok-pok-cookbook.html https://www.serious
eats.com/recipes/2012/12/prosciuotto-and-rosemary-cornish-game-hen-beans-r
ecipe.html
gan https://www.seriouseats.com/recipes/2015/03/20150305-sichuan-dry-fried
-green-bean-broiled-food-lab-recipe.html
ganache https://www.seriouseats.com/recipes/2007/10/how-to-make-macarons-r
ecipe.html  https://www.seriouseats.com/recipes/2019/07/chocolate-american
-buttercream-recipe.html    https://www.seriouseats.com/recipes/2011/10/fr
ench-macaron-recipe-how-to-make-macarons.html
gandules    https://www.seriouseats.com/recipes/2010/10/dinner-tonight-ric
e-with-pigeon-peas-how-to-make-arroz-con-gandules.html
gangnang    https://www.seriouseats.com/recipes/2018/12/hobak-beombeok-kor
ean-squash-sweet-potato-and-bean-porridge.html
ganjang https://www.seriouseats.com/recipes/2019/08/korean-chilled-cucumbe
r-soup-oi-naengguk-recipe.html https://www.seriouseats.com/recipes/2019/01
/doenjang-jjigae-korean-fermented-bean-paste-stew.html https://www.serious
eats.com/recipes/2020/01/korean-anchovy-knife-cut-noodle-soup-myeolchi-kal
-guksu.html https://www.seriouseats.com/recipes/2020/03/miyeok-guk-korean-
seaweed-and-brisket-soup.html     https://www.seriouseats.com/recipes/2019
/05/japchae-recipe.html
ganoush https://www.seriouseats.com/recipes/2014/02/the-best-baba-ganoush-
recipe.html
```

Figure 0: Partial printout of inverted index.

Elastic Enterprise App Search with ReactJS

This involved uploading our preprocessed JSON recipe data to the cloud, which allowed for automatic indexing of documents. The cloud service provided an easy web interface for viewing individual documents, as well as other features like changing the schema for the entire dataset with just a few clicks. In addition, it enabled fast and secure queries over the web, and provided an intuitive npm client for issuing queries and neatly-displaying results with ReactJS.



Engine Overview

Total Queries 45 | Total Documents 2,537 | Total Clicks 4

Total Queries — Last 7 days

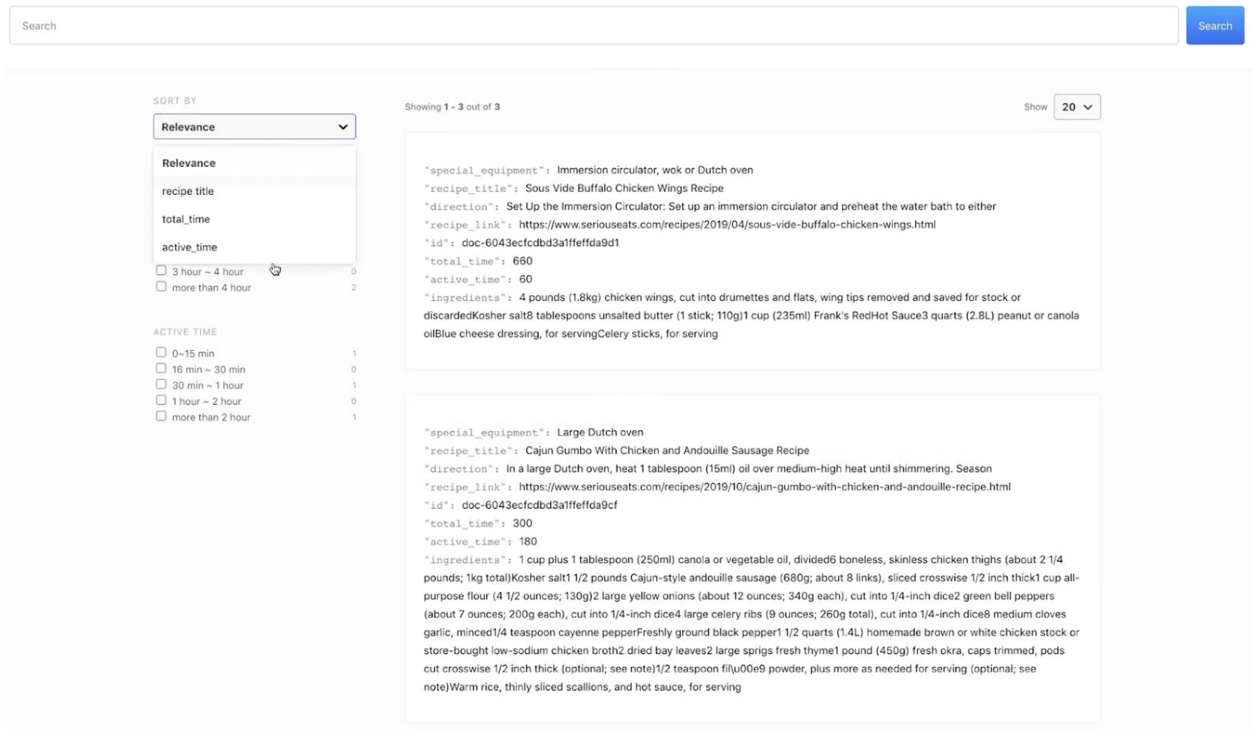
Total Operations — Last 7 days

Recent API Events

Method	Time	Endpoint	Status	
POST	9 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details
POST	9 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details
POST	22 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details
POST	22 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details
POST	43 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details
POST	43 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details
POST	43 minutes ago	/api/as/v1/engines/recipes/search.json	● 200	Details

Figure 1: Search Engine UI provided from Elastic Cloud.

The React application was bootstrapped with the `create-react-app` command line interface. Furthermore we used the Elasticsearch npm client to connect to the remote endpoint, allow for filtering of recipe results by active times and total times, and sort the results by relevance scores, or by the active and total times. Elasticsearch, by default, uses BM-25 for ranking.



```
"special_equipment": Immersion circulator, wok or Dutch oven
"recipe_title": Sous Vide Buffalo Chicken Wings Recipe
"direction": Set Up the Immersion Circulator: Set up an immersion circulator and preheat the water bath to either
"recipe_link": https://www.seriouseats.com/recipes/2019/04/sous-vide-buffalo-chicken-wings.html
"id": doc-6043ecfcdbd3a1ffeffda9d1
"total_time": 660
"active_time": 60
"ingredients": 4 pounds (1.8kg) chicken wings, cut into drumettes and flats, wing tips removed and saved for stock or
discardedKosher salt8 tablespoons unsalted butter (1 stick; 110g)1 cup (235ml) Frank's RedHot Sauce3 quarts (2.8L) peanut or canola
oilBlue cheese dressing, for servingCelery sticks, for serving
```

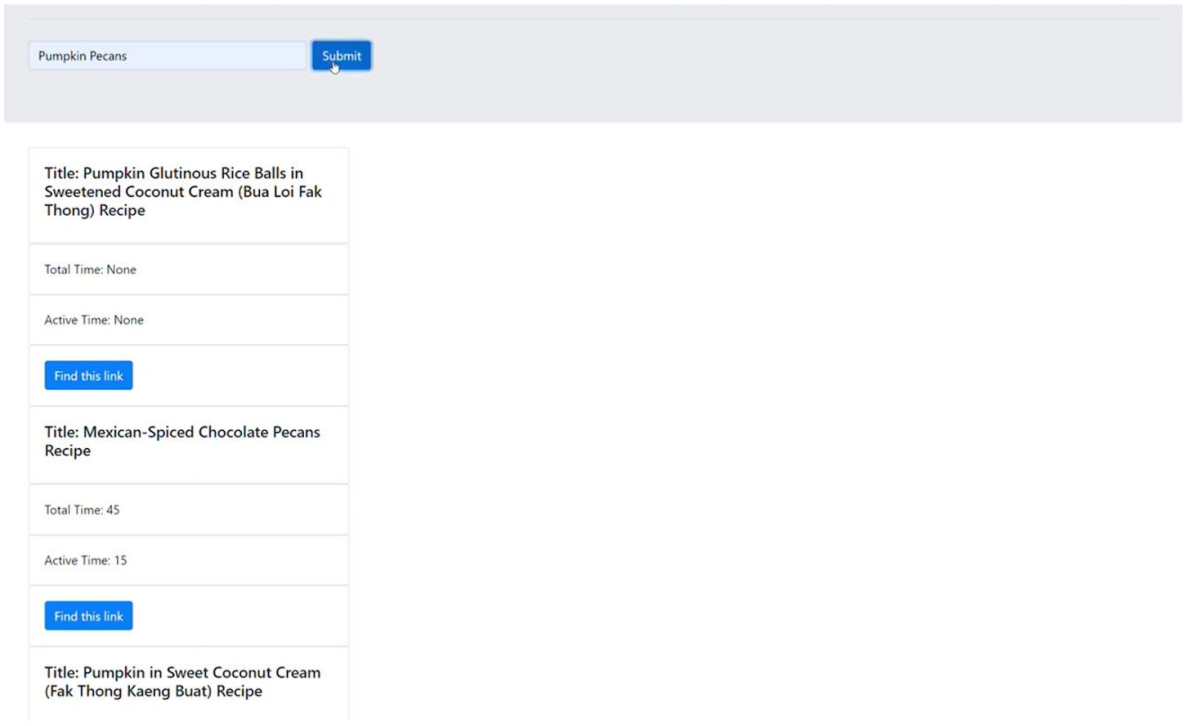
```
"special_equipment": Large Dutch oven
"recipe_title": Cajun Gumbo With Chicken and Andouille Sausage Recipe
"direction": In a large Dutch oven, heat 1 tablespoon (15ml) oil over medium-high heat until shimmering. Season
"recipe_link": https://www.seriouseats.com/recipes/2019/10/cajun-gumbo-with-chicken-and-andouille-recipe.html
"id": doc-6043ecfcdbd3a1ffeffda9cf
"total_time": 300
"active_time": 180
"ingredients": 1 cup plus 1 tablespoon (250ml) canola or vegetable oil, divided6 boneless, skinless chicken thighs (about 2 1/4
pounds; 1kg total)Kosher salt1 1/2 pounds Cajun-style andouille sausage (680g; about 8 links), sliced crosswise 1/2 inch thick1 cup all-
purpose flour (4 1/2 ounces; 130g)2 large yellow onions (about 12 ounces; 340g each), cut into 1/4-inch dice2 green bell peppers
(about 7 ounces; 200g each), cut into 1/4-inch dice4 large celery ribs (9 ounces; 260g total), cut into 1/4-inch dice8 medium cloves
garlic, minced1/4 teaspoon cayenne pepperFreshly ground black pepper1 1/2 quarts (1.4L) homemade brown or white chicken stock or
store-bought low-sodium chicken broth2 dried bay leaves2 large sprigs fresh thyme1 pound (450g) fresh okra, caps trimmed, pods
cut crosswise 1/2 inch thick (optional; see note)1/2 teaspoon fil\u00e9 powder, plus more as needed for serving (optional; see
note)Warm rice, thinly sliced scallions, and hot sauce, for serving
```

Figure 2: ReactJS Search Engine UI showing results, and ability to filter/sort results.

Apache Cassandra with Express/NodeJS

We setup Apache Cassandra on an Ubuntu 18.04.5 virtual machine, loaded the index into a table, and finally, loaded the entire dataset of documents into another table. We were able to use the Cassandra command line interface to load data into a table from a .csv file. Our index.csv consists of two columns, (1) term, (2) list of links where term is found; the term is the primary key of this table. The document dataset.csv consists of all of the fields we chose to store per recipe document, and the primary key of this table is the recipe link.

After instantiating the database, it was time to connect to it using the Express/NodeJS application. Luckily, the npm library provided a Cassandra driver, where we learned how to connect to the database, and perform queries. The connection to the database was the easy part, as we were able to console.log hardcoded queries from the node.js application. The difficult part was to design a web UI displaying a search box and submit button and serve it using Express and NodeJS. Furthermore, we spend a lot of time figuring out how to take query results and display them in a presentable fashion on the front end.


Pumpkin Pecans

Submit

Title: Pumpkin Glutinous Rice Balls in Sweetened Coconut Cream (Bua Loi Fak Thong) Recipe

Total Time: None

Active Time: None

Find this link

Title: Mexican-Spiced Chocolate Pecans Recipe

Total Time: 45

Active Time: 15

Find this link

Title: Pumpkin in Sweet Coconut Cream (Fak Thong Kaeng Buat) Recipe

Figure 4: Screenshot of Express/NodeJS interface after searching for "Pumpkin Pecans", demo of screenshot here:

https://drive.google.com/file/d/1RdDPGTuXpGI7y9YxKI_cBGtXo9Rq68Za/view?usp=sharing

System Limitations

Limitations regarding the Hadoop Search Engine consisted of choice of ranking due to much more time invested in incorporating MapReduce-generated index into our Cassandra Database and connecting the DB and backend server into our frontend. So we were not able to fully implement an ideal ranking system. We struggled so much with uploading our full dataset into Cassandra, since we had to spend a lot of overhead figuring out how to change ~2500 documents from JSON data to csv. Due to issues such as delimiting data by commas, or strings being excessively long, some of the documents were not added to Cassandra, as Cassandra would fail on these instances.

The bulk of the time was spent on learning and implementing a full-stack web development. Once all of the pieces were connected together (i.e. back end, middle end, front end), we finally tackled results displaying, as well as results ranking. Every keyword in our query needed to be queried, and we did not have time to implement a more effective method of querying such as batch queries. For every user query, it required multiple queries, one per query keyword. Although the method is easily explainable, it is unfortunate that it is also highly inefficient. Furthermore, the ranking is very basic, as we simply just perform the intersection of documents as a method of ranking between search results.

One other important limitation is how we obtain the set of intersected documents. Since we did not have time to figure out how to query efficiently, we simply decided to query Cassandra for every document and the intersected list. This turned out to be highly inefficient, though simple in design and implementation.

Obstacles and Solutions

We struggled to choose a ranking algorithm, especially leaning towards BM-25 due to its consideration of document and query term weights. Ultimately, we chose to use a non-stemmed intersection-based ranking system because we agreed that the most common use case is searches by ingredients, many of which are foreign words. For example, we expect poor query results if “ricer” were to be stemmed to “rice” (the former referring to a kitchen tool, the latter referring to the popular grain). Furthermore, we concluded that the common case of the singular and plural was a nonissue since a recipe with an ingredient in the title would often refer to it in the opposite form. For example, “Tomato Soup” would surely include “tomatoes” in the ingredients and directions while the same would be true of “Refried Beans” and “bean”, negating the benefits of stemming and avoiding the complications of stemming foreign words.

Deployment Instructions

Please refer to the following text documents located inside the root directory of the repository linked at the top of this document.

- Installing-Cassandra-on-Ubuntu-18.04.5.txt
- ejs-mrsearch/instructions.txt

For Hadoop and MapReduce, please use Maven to run the files included in the Hadoop_MapReduce directory. We have been running it using the following command lines:

```
rm output; mvn package && hadoop jar
target/MapReduce-1.0-SNAPSHOT.jar
edu.ucr.cs.cs242.RecipeSearch.MapReduce
```

References

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

https://timepasstechies.com/map-reduce-inverted-index-sample/

https://codeburst.io/how-to-build-great-react-search-experiences-quickly-8c69081f328d

https://cassandra.apache.org/doc/latest/getting_started/installing.html

https://www.npmjs.com/package/cassandra-driver https://github.com/DataStax-Examples/quickstart-nodejs/blob/master/quickstart.js

https://sundog-education.com/elasticsearch/

https://www.udemy.com/elasticsearch-6-and-elastic-stack-in-depth-and-hands-on/

https://www.elastic.co/blog/how-to-find-and-remove-duplicate-documents-in-elasticsearch

https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/api-reference.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-all-query.html

http://exploringelasticsearch.com/searching_data.html#ch-searching-data

http://okfnlabs.org/blog/2013/07/01/elasticsearch-query-tutorial.html#query-dsl-overview

https://www.youtube.com/watch?v=52G5ZzE0XpY#t=1471

https://sundog-education.com/elasticsearch/

https://towardsdatascience.com/getting-started-with-elasticsearch-in-python-c3598e718380

https://www.elastic.co/guide/en/elasticsearch/guide/current/_talking_to_elasticsearch.html

https://www.elastic.co/guide/en/elasticsearch/client/index.html

https://stackoverflow.com/questions/24153996/is-there-a-limit-on-the-size-of-a-string-in-json-wit
h-node-js

https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-delete-index.html

https://www.elastic.co/guide/en/elasticsearch/reference/6.1/_list_all_indices.html

Tutorial on Webscriping and parsing with BeautifulSoup

https://stackoverflow.com/questions/1936466/beautifulsoup-grab-visible-webpage-text

https://chartio.com/resources/tutorials/how-to-install-elasticsearch-on-mac-os-x/

https://stackoverflow.com/questions/12451997/beautifulsoup-gettext-from-between-p-not-pickin
g-up-subsequent-paragraphs

https://stackoverflow.com/questions/50045253/if-statement-in-ejs

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query.html

https://stackoverflow.com/questions/10326950/render-a-variable-as-html-in-ejs

https://stackoverflow.com/questions/50103466/ejs-cannot-read-property-of-undefined-how-can-i
-fix-this

https://code.tutsplus.com/articles/introduction-to-parallel-and-concurrent-programming-in-python
--cms-28612

https://docs.python.org/2/library/threading.html

https://www.oreilly.com/learning/python-cookbook-concurrency

https://youtu.be/gnsO8-xJ8rs

https://getbootstrap.com