

# Part A: Collect Your Data and Index with Lucene

## Repository

<https://github.com/rmace001/RecipeSearchEngine>

## Team Members

Rogelio Macedo - [rmace001@ucr.edu](mailto:rmace001@ucr.edu)

Jerry Tan - [jt看027@ucr.edu](mailto:jt看027@ucr.edu)

Erin Wong DeForest - [ewong012@ucr.edu](mailto:ewong012@ucr.edu)

Shiyao Feng - [sfeng023@ucr.edu](mailto:sfeng023@ucr.edu)

## Collaboration Details

### Crawler

Jerry Tan

Worked on developing the web scraper using Python via BeautifulSoup in order to extract meaningful and relevant information regarding recipes.

Shiyao Feng

Worked on developing the web scraper using BeautifulSoup in Python to download the HTML page and extract any hyperlinks in the seed page.

### Indexer

Rogelio Macedo

Research and application of instantiation of virtual Elasticsearch clusters. Performed and established the final data formatting and preprocessing of the data so it can be fed to Elasticsearch.

Erin Wong DeForest

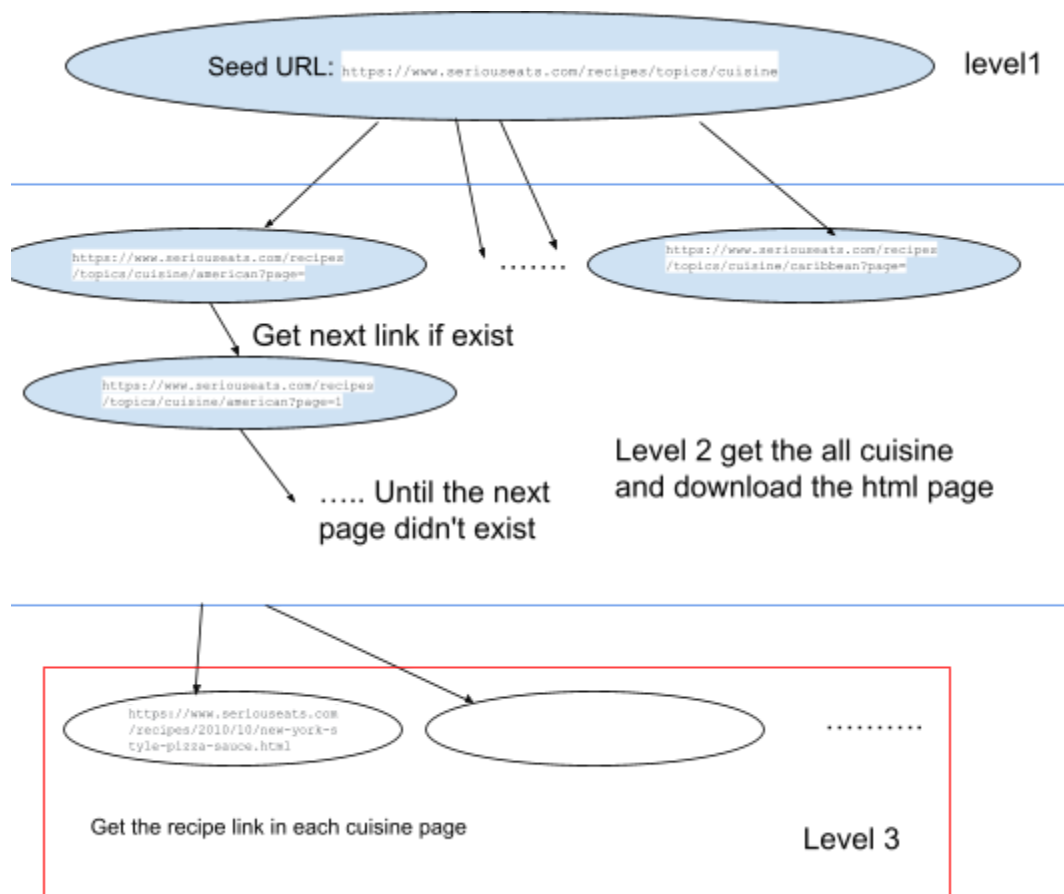
Research and application of Elasticsearch functionalities such as querying. Performed creation of the index from pre-processed data and experimented with retrieving results for different query types such as by ingredient or recipe title.

# Crawler

## Overview

The main idea of Web Crawler is a program that, gets one or more seed HTML link, downloads the web pages related to the seed URL, extracts any hyperlinks or content contained in the seed page, and recursively continues to find the next page link and download the next HTML identified by the hyperlinks.

## Architecture



## Crawling Strategy

After conducting some research, we determined that the best website containing recipe information is [Serious Eats](#), a site in which multiple recipes can be shared amongst food enthusiasts.

The information we wanted to extract from each recipe is as follows

- Recipe link
- Recipe title
- Special equipment

- Notes
- Total Time
- Active Time
- Ingredients
- Directions

In order to minimize the potential number of duplicate recipes and pages, we decided to scrap all the recipes based on cuisine. There are 19 cuisines on Serious Eats, and we decided to develop our scraper such that it has an array of 19 links of cuisines and then crawl all the corresponding recipes under those cuisines.

## Steps to Produce

1. Create 'data' folder to hold all level data
2. To run the crawler execute the command `python3 scraper.py`
3. After some time the crawler will generate three levels of data, which will contain all the raw html files of recipes (code will be automated to build those three level folders)
4. Data.json is where all the recipes relevant information will be located to be used for indexing

## Obstacles and Solutions

We had a bit of a challenge figuring out how our scraper would retrieve recipe links on a cuisine page, because from a HTML perspective, there was no distinct way to retrieve recipe links without retrieving unnecessary information. However, we were able to retrieve the latest `<script type="application/ld+json">` which contains all the recipe urls that we wanted to extract.

Runtime, Level & total memory:

```
web crawling spend (time.time): 569.7486040592194
web crawling spend (time.clock): 569.747325385
web crawling reach memory: 677390240
web crawling reach page: 2651
```

Here is 677390240 bit = 677 MB > 250 MB

## Resources

Main source of data: [Serious Eats](#)  
[beautifulsoup Documentation](#)

# Indexer

## Overview

We opted to use Elasticsearch rather than just Lucene because it gives us the functionality of Lucene and the convenience of REST API commands. Other than the practicality of learning how to work with virtual clusters, it was beneficial to take this route given Elasticsearch provides a distributed system over Lucene.

There are three main pieces to highlight. The first is the preprocessing necessary in order for the data to be recognizable for Elasticsearch. The second is the virtual cluster setup. Lastly, we have the creation of the Elasticsearch index. Each of these steps take ~5-10 seconds to execute.

## Fields in Index



## Text Analyzer Choices

We chose to use the standard Elasticsearch analyzer because of its generality and usefulness for most languages. We expect most recipes to be overwhelmingly English words but with various terms and ingredients from other languages. So far, the standard analyzer has been satisfactory, but if we realize during part B that it must be adjusted, we will build a custom analyzer using the standard analyzer as a jumping-off point.

## Run Time of Index Creation Process

Indexing 2537 objects takes less than one minute to complete.

```
user@ubuntu:~/RecipeSearchEngine$ time curl -H 'Content-Type: application/json' -X POST 'localhost:9200/recipes/doc/_bulk?pretty' --data-binary @ofile.json > index_timing.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 6450k  100  892k  100 5557k   1043k   6492k   --:--:--   --:--:--   --:--:--  7526k

real    0m0.878s
user    0m0.011s
sys     0m0.018s
```

## Steps to Produce

### Data Preprocessing

In order to use a curl command to bulk-load json data into a brand new index, the data JSON file must be formatted in a special way. Furthermore, the JSON file must end with a new line. Here is an example of a valid JSON file which can be indexed by Elasticsearch:

```
{"create": {"_index": "recipes", "_type": "rwebdoc", "_id": "0"}}
{"id": "0", "recipeLink": "<link>", "recipeTitle": "Title",
"specialEquipment": "tool", "notes": "", "activeTime": "3 hours",
"totalTime": "5 hours", "direction": "", "ingredients": ""}
```

*Example data.json*

Above we see data.json; it consists of only three lines, yet a total of one document. The first line is the creation object, the second line is the web document object, and the last line is just an empty line. The create object is needed to let Elasticsearch know where we will be placing the web document. In other words, in the create line we define the index which will contain the document. We also define the type of document we want to index, and lastly the document ID. The value for `_index` and `_type` are user-defined, and can be uniquely named to be anything. Note that all documents must be of the same type for a particular index. Also note that each document will have its own create line. If we were to have two documents in data.json, then we would see the create object for document one, followed by the actual document one object, followed by the create object for document two, followed by the actual document two object, followed by a newline. Each pair of objects (one pair for document one, another for document two) must contain the same value for the ID field. Each object must be on a single line.

### Virtual Cluster Setup

- We use a virtual machine created on VMware Workstation 16 Player
- Install Ubuntu 18.04.5 (16.04.7 would not allow elasticsearch to work)
- Install OpenJDK JDK 8 and JRE 8
- Install Elasticsearch 6.8.13 and ensure network host set to 0.0.0.0

```
# Ubuntu 18.04 BIONIC BEAVER
sudo apt-get install openjdk-8-jre-headless -y
sudo apt-get install openjdk-8-jdk-headless -y
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -
sudo apt-get install apt-transport-https
echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main"
| sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list
```

```
sudo apt-get update && sudo apt-get install elasticsearch
sudo vim /etc/elasticsearch/elasticsearch.yml
# EDIT AND UNCOMMENT network.host: 0.0.0.0
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable elasticsearch.service
sudo /bin/systemctl start elasticsearch.service
sudo apt install curl
# Check Elasticsearch
curl 127.0.0.1:9200
```

## Index Creation

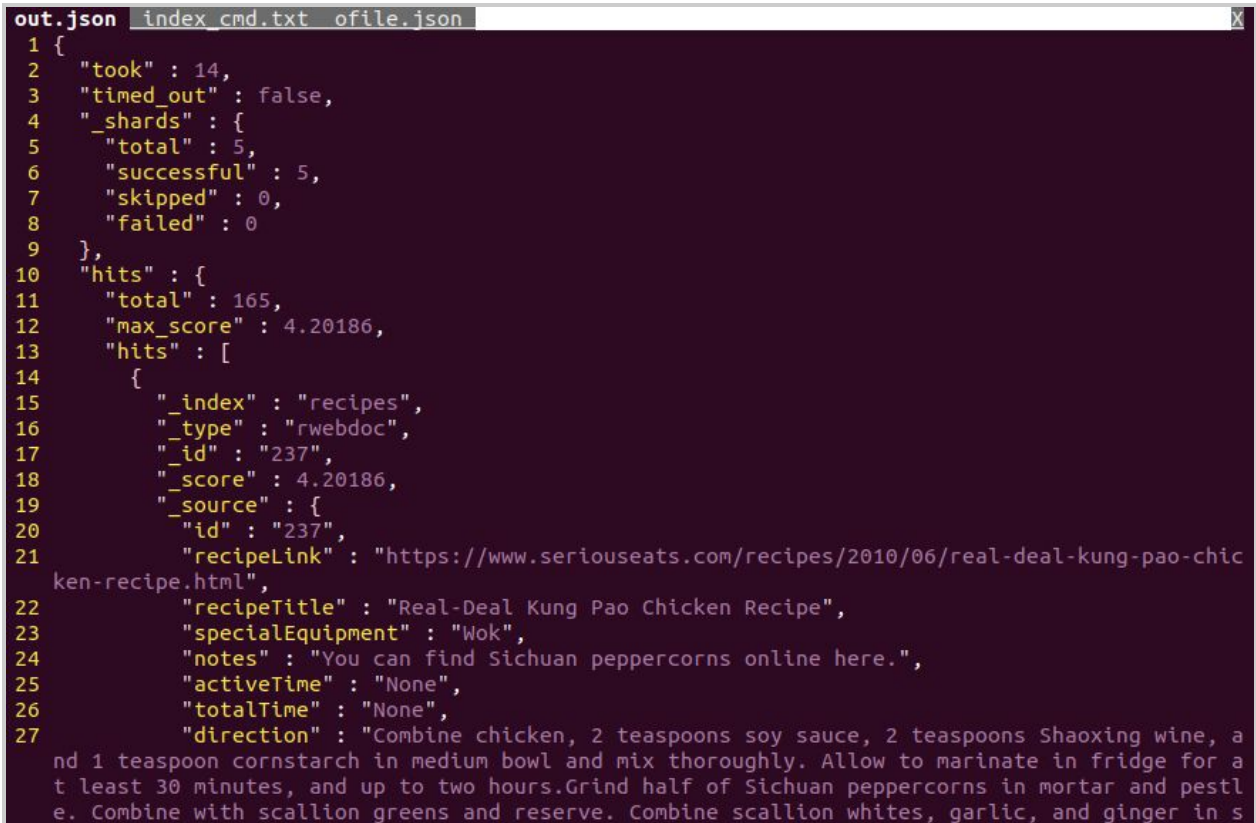
Ensure bulk data JSON file ends with a newline. Index the file using POST.

```
curl -H 'Content-Type: application/json' -X POST
'localhost:9200/recipes/doc/_bulk?pretty' --data-binary @ofile.json
# View indices
curl -X GET 127.0.0.1:9200/_cat/indices?v
# Ugly search (don't use)
curl -X GET localhost:9200/_search?q=chinese
# General query
curl -H "Content-Type: application/json" -XGET
'127.0.0.1:9200/recipes/_search?pretty' -d '
{
  "query": {
    "query_string" : {
      "query" : "mediterranean"
    }
  }
}' > out.json

# Delete index:
curl -X DELETE 127.0.0.1:9200/recipes

# Query field
curl -H "Content-Type: application/json" -XGET
'127.0.0.1:9200/recipes/_search?pretty' -d '
{
  "query": {
```

```
"term" : {  
  "specialEquipment" : "wok"  
}  
}  
' > out.json
```



```
out.json index_cmd.txt ofile.json  
1 {  
2   "took" : 14,  
3   "timed_out" : false,  
4   "_shards" : {  
5     "total" : 5,  
6     "successful" : 5,  
7     "skipped" : 0,  
8     "failed" : 0  
9   },  
10  "hits" : {  
11    "total" : 165,  
12    "max_score" : 4.20186,  
13    "hits" : [  
14      {  
15        "_index" : "recipes",  
16        "_type" : "rwebdoc",  
17        "_id" : "237",  
18        "_score" : 4.20186,  
19        "_source" : {  
20          "id" : "237",  
21          "recipeLink" : "https://www.seriousseats.com/recipes/2010/06/real-deal-kung-pao-chic  
ken-recipe.html",  
22          "recipeTitle" : "Real-Deal Kung Pao Chicken Recipe",  
23          "specialEquipment" : "Wok",  
24          "notes" : "You can find Sichuan peppercorns online here.",  
25          "activeTime" : "None",  
26          "totalTime" : "None",  
27          "direction" : "Combine chicken, 2 teaspoons soy sauce, 2 teaspoons Shaoxing wine, a  
nd 1 teaspoon cornstarch in medium bowl and mix thoroughly. Allow to marinate in fridge for a  
t least 30 minutes, and up to two hours.Grind half of Sichuan peppercorns in mortar and pestl  
e. Combine with scallion greens and reserve. Combine scallion whites, garlic, and ginger in s
```

*Result of searching for “wok” in the “specialEquipment” field*

## Limitations

At this time, we are able to query by general terms of interest, ingredients, and special equipment (such as “wok”). However, we are currently unable to query based on the time required by the recipe. Ultimately, we aim to allow users to search for recipes which are do-able within a certain length of time.

## Obstacles and Solutions

One of the major obstacles we experienced was the deployment of the virtual cluster. Our first challenge was selecting an appropriate development environment. Initially, we struggled greatly to get Elasticsearch set up; this turned out to be due to the version of Ubuntu we were initially using: 16.04. Upgrading to Ubuntu 18.04 resolved this issue. Additionally, while attempting to solve the issue we rolled back from the latest release of Elasticsearch to v6.8.

After we got Elasticsearch up and running, we had to figure out how to create an index and populate it with our data. We found that in order to load an entire file of parsed web data, the data JSON file needs to be formatted correctly. To do this, we created a script to generate the creation object and insert newlines where appropriate.

## Resources

[VMWare Workstation 16 Player](#)

[Ubuntu 18.04.5](#)

[Elasticsearch 6.8.13 Documentation](#)

[Document APIs | Elasticsearch Reference \[6.8\]](#)

[Built-in Elasticsearch analyzers reference](#)

[Standard Elasticsearch analyzer](#)

[Curl Syntax In Elasticsearch With Examples | Curl Commands](#)

[What is the difference between Lucene and Elasticsearch](#)

[Elasticsearch Tutorial: Creating an Elasticsearch cluster](#)