**ResearchDocs –** Search Engine for research papers

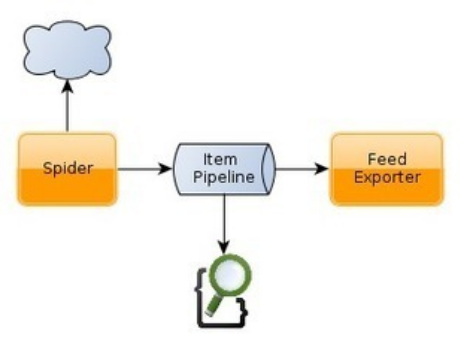
Puneeth Shankar Bikkasandra Puttaramegowda

**1 Motivation**

We come across manifold number of websites serving research papers on the internet. But does all of them really serve the need of the user? To combat this problem with accuracy, I have built a search engine which recommends latest research papers in the Computer Science research Community. The source of this search platform is the well-known portal ***arXiv.org*** from Cornell University. The user may choose to search the paper based on the term, authors, title and subject (area of research).

**2 System Design**

**2.1 Architecture**



Crawler (scrapy) would iterate through the seed list and scrape the necessary fields mentioned in the scraping and search function and format using pipelining and perform indexing as described in the below sections of this document.

**2.2 Infrastructure**

|  |  |
| --- | --- |
| **Hardware** | **Software** |
| **Virtual Machine**:  2 cores,  RAM: 6GB,  Memory: 60GB  **Personal Laptop**: MacBook Pro  Processor: Intel Core i5.  RAM: 4GB.  Memory: 8GB. | Crawling **: Scrapy**  version – 1.6  Distributed DB **: ElasticSearch**  version – 6.6.1  Data Source***: arXiv.org***  Seed Extraction**:** python libraries **(***BeautifulSoup***)**  Json Formatting : **ScrapyElasticSearch**  Search Engine Analytics :  **Kibana**  Version – 7.0 |

**3 Implementation**

Software Installations:

1. ElasticSearch 6.6.1

wget <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.0.0-linux-x86_64.tar.gz>

Unzip and run elasticsearch under *bin* directory.

1. Kibana 7.0

wget <https://artifacts.elastic.co/downloads/kibana/kibana-7.0.0-linux-x86_64.tar.gz>

Unzip and run kibana under *bin* directory.

1. ScrapyElasticSearch

pip install “ScrapyElasticSearch”

**3.1 Web Crawler**

To crawl the list of web pages, I have to start from any page from *arXiv.org* similar to the one in *Figure 1*.

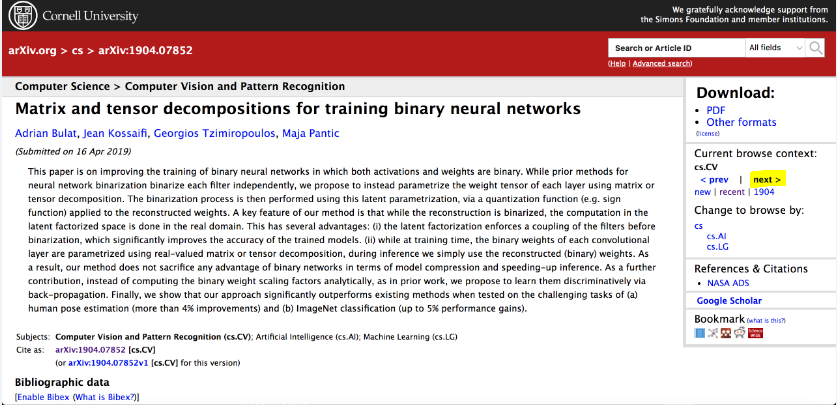


Figure 1

The below mentioned crawler (in python in Figure 2) would scrape following fields from the above page:

* Title
* Abstract
* Authors
* PDF link
* Submission Date

Apart from scraping the above fields and appending it to the json content, crawler would navigate through the ‘NEXT’ link (highlighted in yellow) found on the right side of the page (as below in Figure 1).



**Figure 2**

**3.1.1 Seed URLs**

I managed to obtain around 3000 unique URLs by running the crawler with some configuration management mentioned below.

**3.1.2 Crawl Policy**

I had to make below modifications to crawl *arXiv.org* due to the robots.txt exclusion.

1. Disobey the *robots.txt* in the scrapy setup.
2. Increase the delay time to 20 seconds as mentioned by the *robots.txt* instructions.
3. Add the User agent, in case I have been crawling illegimately.

Post obtaining the seed URLs, we are supposed to index the scraped data, which is achieved through the use of a tool “ScrapyElasticSearch”, performing item pipelining. Indexing of the documents by creating the json data of specific format and feeding it to *kibana* is the other way achieving the same.

Explicitly, we do not have to perform any information retrieval tasks such as (Stemming, Lemmatization. Etc) as it is taken care by Elasticsearch. The below changes were made to the *settings.py* found on the scrapy installation module, where ‘**research’** is my index name and ‘**doc’** is the type.

ITEM\_PIPELINES = {

'scrapyelasticsearch.scrapyelasticsearch.ElasticSearchPipeline':100,

}

ELASTICSEARCH\_SERVER = 'localhost'

ELASTICSEARCH\_PORT = 9200

ELASTICSEARCH\_INDEX = ‘research’

ELASTICSEARCH\_TYPE = ‘doc’

ELASTICSEARCH\_UNIQ\_KEY = 'title'

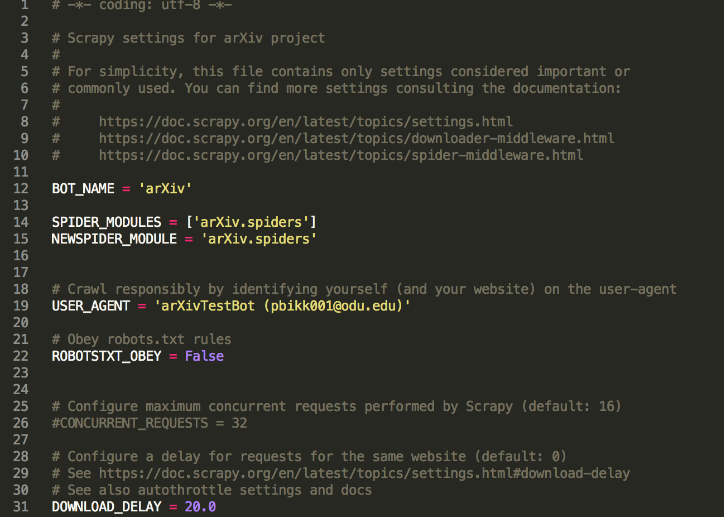


Figure 3

**3.1.3 User Interface:**

User Interface (UI) was created using HTML/CSS and javaScript. Through port forwarding, I was able to hit the below URL in UI code and page was up by running the HTML file locally.

URL used in code:

"http://localhost:9200/research/doc/\_search?q="+criteria+":"+'"'+searchText+'"'+"&size=1000"

In the above URL, I’m accessing the search API from elasticsearch and reading the user selection at the dropdown html element through ‘criteria’ variable and query term through ‘searchText’.

By default elasticSearch returns only 10 documents for any query term, to increase it, we need to add a parameter ‘size’ to the number we would like to show on the UI. Code is attached at the end of this report.

**Port Forwarding**:

ssh pbikkasa@linux.cs.odu.edu -L 9200:localhost:9200

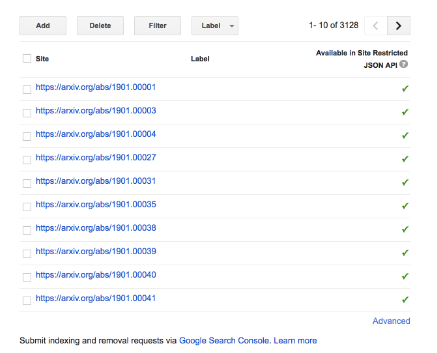
ssh student@host-3.cs734.cs.odu.edu -L 9200:localhost:9200



**4 Google Custom Search**

Google Custom Search can be accessed through below URL:

<https://cse.google.com/cse?cx=001046393167602876589:ysksjhvo8ya>



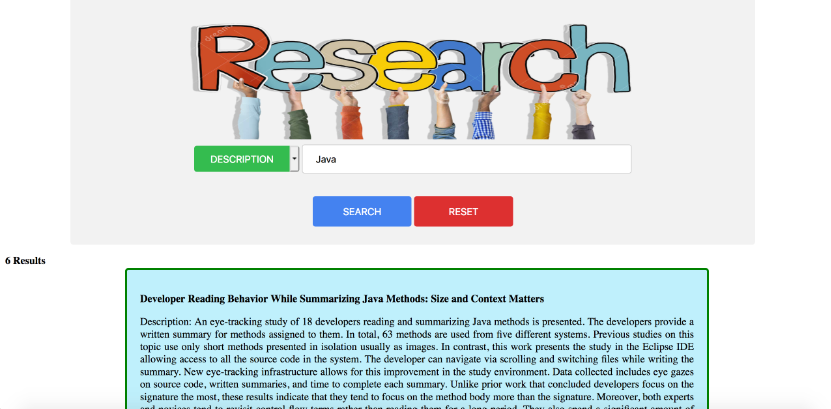
GCE is configured to handle bulk URLs as mentioned above.

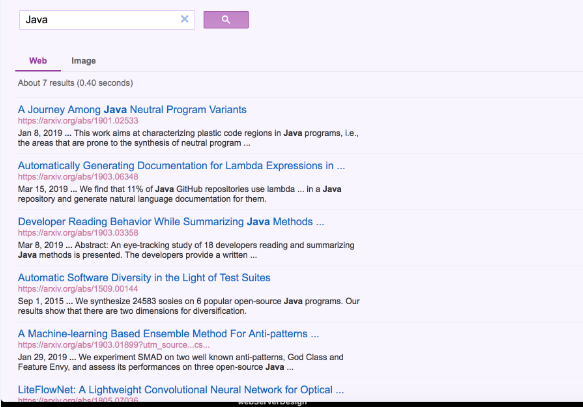
**5 Evaluation and Comparison**

Evaluation of both the search engine were made using Precision@K. I have chosen the K value to be 10.

Same set of seed list are used for both GCE and SSE.

|  |  |  |
| --- | --- | --- |
| **Query** | **P@10** | **GCE** |
| Machine Learning | 0.98 | 1 |
| Computer Vision | 0.89 | 0.98 |
| Pattern Recognition | 0.98 | 1 |
| Java | 0.99 | 1 |





**6. Conclusion:**

It appears that the results obtained through GCE are a bit efficient in terms of relevance, as you can observe the results for the query term ‘Java’ above.

The results obtained on the SSE can be improved if we could search on the fields opted while indexing. Relying on my understanding, people would like to use search engine for research papers when they want to search anything specific, in my case user would be having a prior knowledge of the terms he/she wants to search on the search engine on the title, authors, subject and at least rarely on abstract.

# **7**. **References:**

[1]Elasticsearch. 2019. *Elasticsearch Documentation.* https://www.elastic.co/guide/index.html.

website, Elasticsearch. 2019. *Elasticsearch Documentation.* January 31. https://www.elastic.co/guide/index.html.

[2] Edureka. 2017. *Edureka.* November1 14. https://www.youtube.com/watch?v=1EnvkPf7t6Y.

Elasticsearch. 2019. *Elasticsearch Documentation.* https://www.elastic.co/guide/index.html.

website, Elasticsearch. 2019. *Elasticsearch Documentation.* January 31. https://www.elastic.co/guide/index.html.