OER Commons Search Engine

IST 441: Specialty Search Engine Project Final Report

Nick D., Wyatt N., Yuya O., and Dana S.
Group 8

Outline

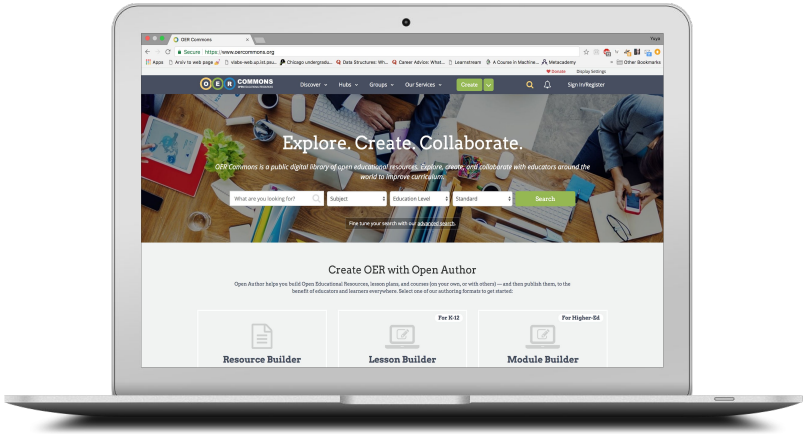
1. Introduction
2. System Architecture
3. Crawler System
4. Document Preprocessing
5. Indexing Methods
6. Front-End Interface
7. Results
8. Conclusion

Introduction

- MOOCs (Massively Online Open Courses) and other open-source and free educational content has recently dominated the internet.
- This has also led to the curation of free content that are now available for online learners to access.
- OER Commons (Open Educational Resources) is a open-source and community driven effort to curate free resources in accessible manner.

However, a *useful speciality search engine* for OER Commons content does not yet exist...

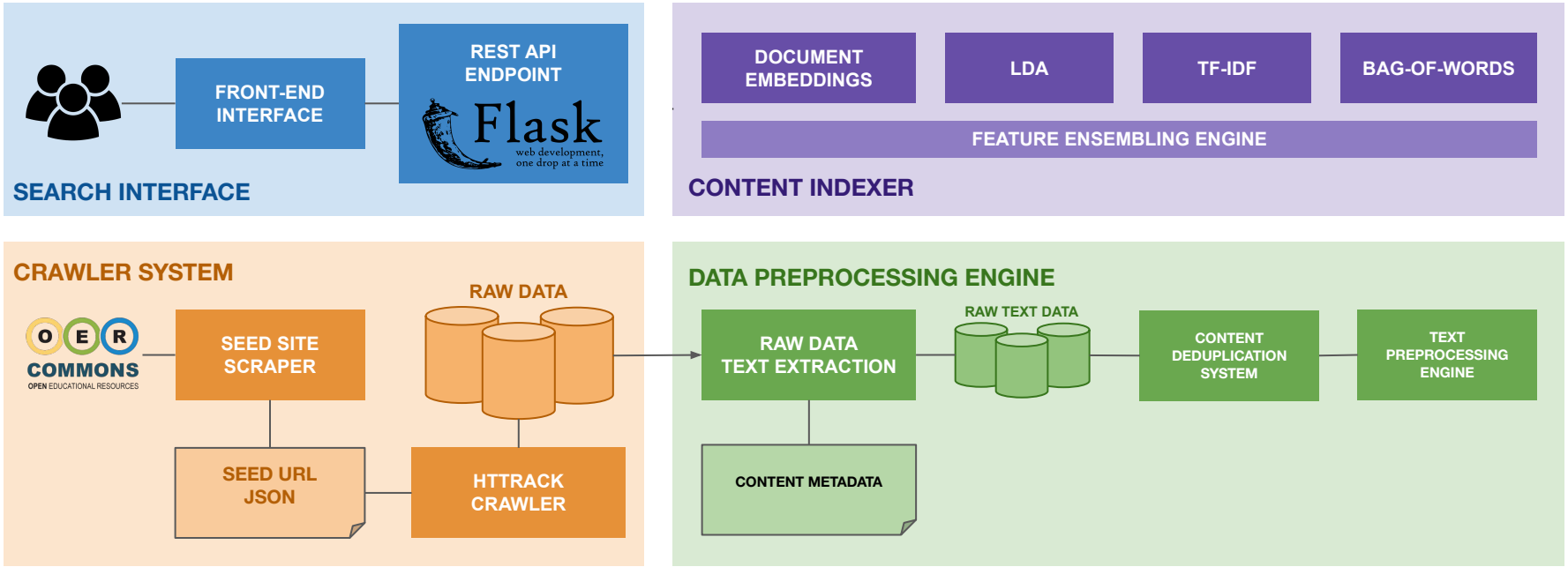
OER COMMONS
OPEN EDUCATIONAL RESOURCES



Customer Information

- Professor Fusco from the College of IST
- Background in Enterprise System Architecture
- Expressed interest in having an OER Search Engine because he believed that “this would be very powerful to implement within a classroom setting.”

System Architecture



Note: We wrote (almost) everything from scratch!

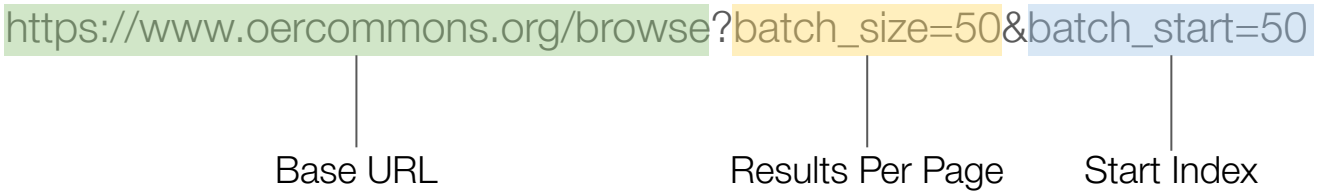
Seed Crawler System

Objective: Obtain list of materials and site content posted on OER Commons Resource.

Discovered blank query leads to display of entire OER Commons Resource content.

Plan: Using Python's **Requests** and **BeautifulSoup** library, we scrape *URLs* and corresponding *metadata* from the OER Commons search page.

OER Query URL Structure



Seed Crawl Statistics

Total Seed URL: 50092

Scheme Distribution

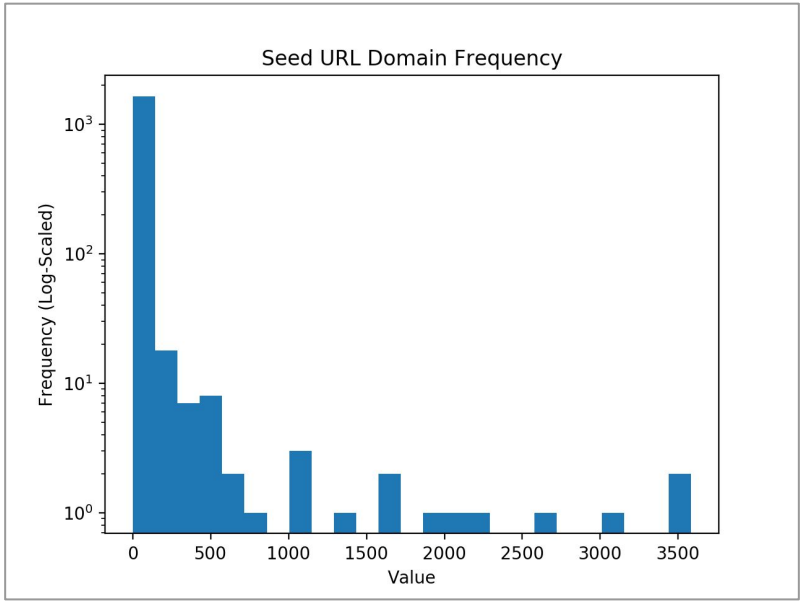
- HTTP: 35292
- HTTPS: 14800

Top 5 URL Domains

1. org - 26903
2. edu - 10032
3. com - 4987
4. gov - 4805
5. ie - 703

Top 10 Domain Freq

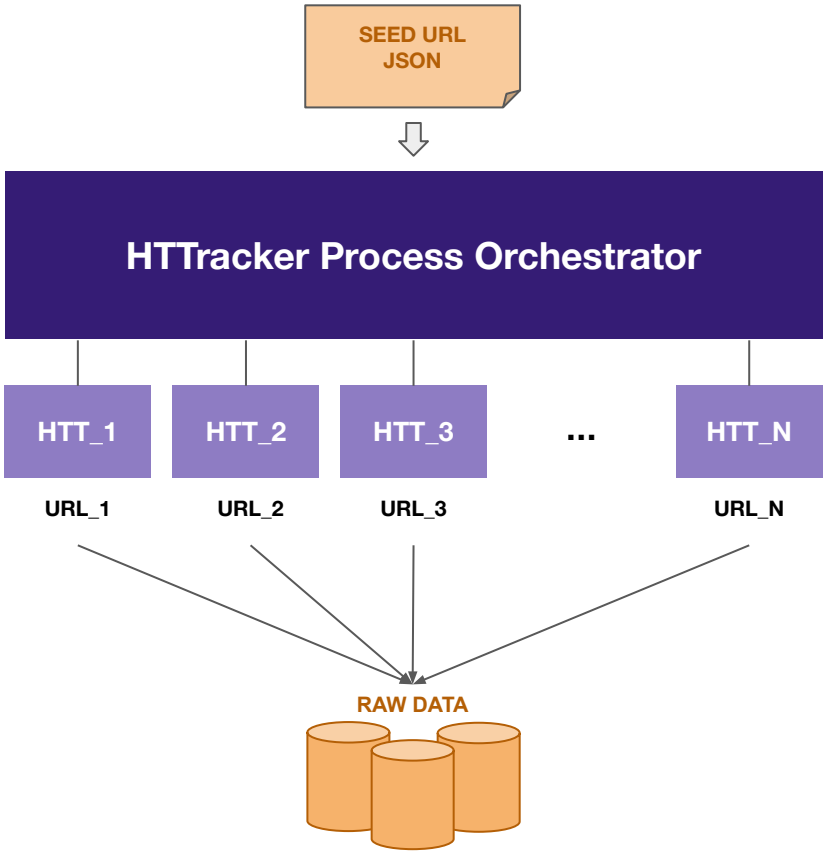
1. cnx - 3582
2. geogebratube - 3517
3. loc - 3152
4. animaldiversity - 2623
5. youtube - 2216
6. carleton - 2144
7. mit - 1981
8. teachengineering - 1630
9. ck12 - 1625
10. archive - 1416



HTTracker Site Crawler

Objective: Download seed site and subsequent link documents - *up to two URL levels deep* and build a raw data repository that we can analyze and index.

- **HTTracker** is an *open-sourced site-mirroring utility* used for site archival system.
- HTTracker is easy to use and highly configurable.
- Parallelization is not so easy, so we wrote our own *multi-process* orchestrator to run multiple instances of HTTracker per seed url.

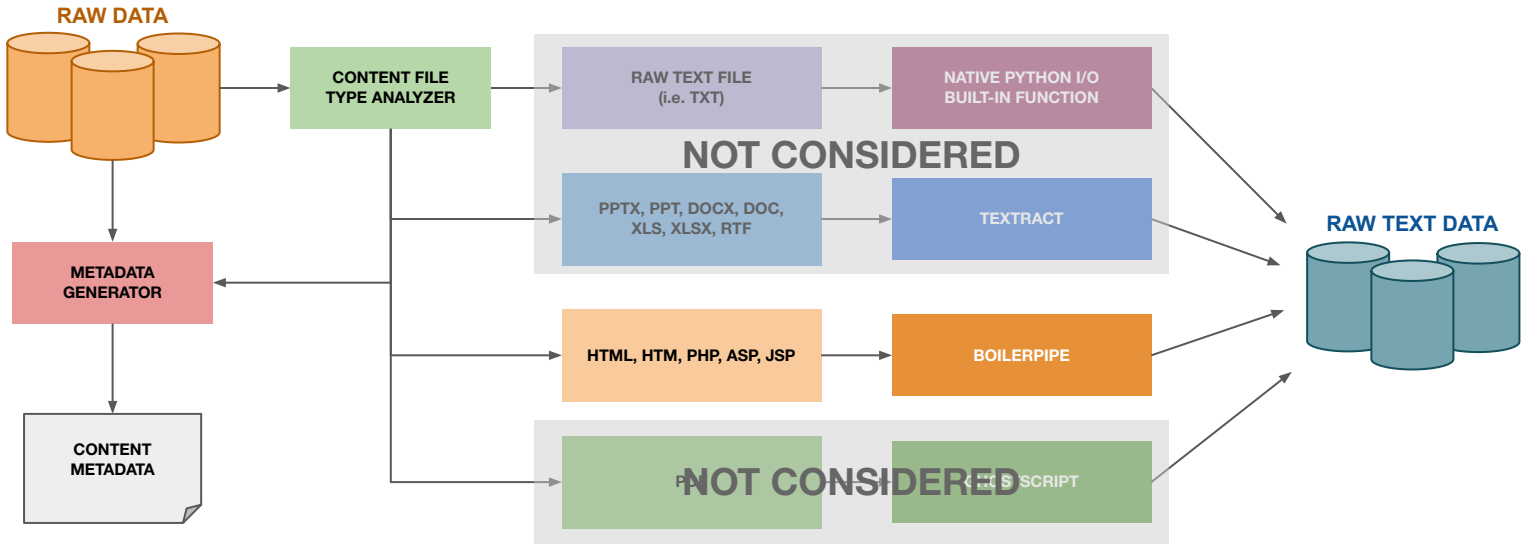


Document Preprocessing

Raw Data Content Extraction Process

Objective: Extract raw text and generate document metadata for each raw data crawled.

For this, we implemented the following **Raw Data Content Extraction** pipeline:



Document Preprocessing

Metadata Generator

For each website, the following metadata are generated:

UUID: Unique document identifier which will be used to normalize naming convention of files.

URL: Original URL of the website scraped.

Title: Website title, based on the title tag (if exist, else use the URL as the title).

All metadata is stored as a json file which will be used in the final web server for quickly mapping the result UUID to the corresponding content.

Document Preprocessing

Boilerpipe Content Extraction

Boilerpipe is a specialized library designed to remove “irrelevant” clutter content.

This package has been shown to work really well in removing things such as navigation bars, headers, footers, and anything that is not really related to the primary content of the site - hence improving overall retrieval performance.

Paper: http://www.l3s.de/~kohlschuetter/boilerplate/

Using Boilerpipe, we filter out any sites that have a ratio based on the following heuristic:

0.1 < Boilerpipe Processed Character Count / Total Document Character Count

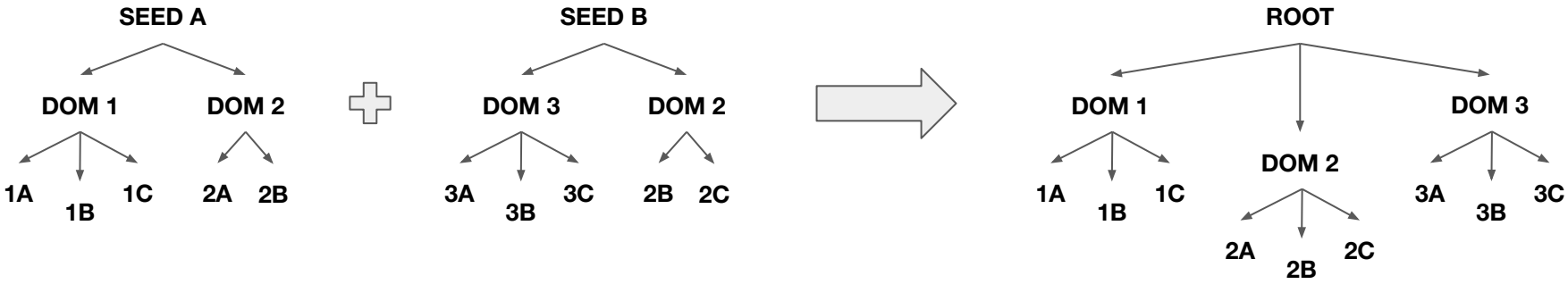
Raw Data Content Deduplication

Problem Description

Since we ran a single HTTrack instance per seed site, there may be potential content redundancies we must consider.

Objective: Develop a deduplication routine to consolidate the raw data hierarchy into a single unified tree structure such that no page duplicates exists.

Example:



Raw Data Content Deduplication

Algorithm Strategy

To deduplicate (or merge) the individual file tree structure into a single unified tree data structure, we implemented the following algorithm:

1. Serialize the directory tree into a single list representation.
2. Merge the lists together with a hashmap.
3. Deserialize the final merged list and construct unified tree file structure.

Reduction Results:

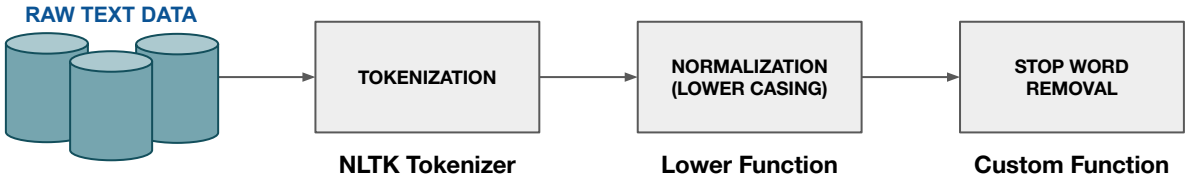
Total Scraped: 397,848 pages Pre-Redupe Website Count: 307236 pages

Post-Processed Website Count: 109,650 pages

Document Preprocessing

NLP Preprocessing Routines

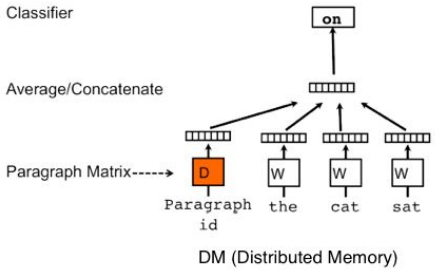
In our preprocessing routine, we have implemented the following pipeline:



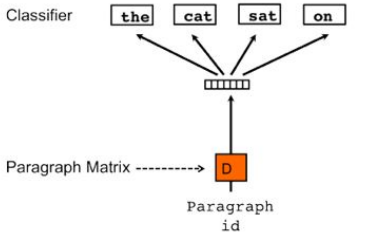
Document Vectorization

Document Embeddings (Doc2Vec)

- Method proposed by Le et. al to convert variable documents or lengths of text into a single vector representation.
- Utilizes a distributed memory model (neural networks) similar to word embeddings (Word2Vec).
- Utilized Gensim to construct the document embeddings.



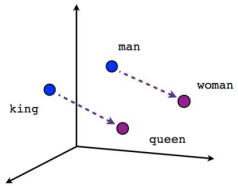
DM (Distributed Memory)



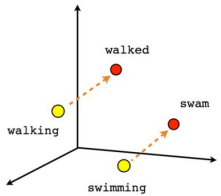
DBOW (Distributed Bag of Words)

Training Process (Hyperparameters):

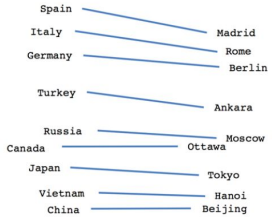
- Method: Skip-Gram Variant
- Vector Dimension: 1024
- Training Epochs: 500
- Window Size: 8
- Minimum Word Count: 5
- Learning Rate (alpha): 0.001



Male-Female

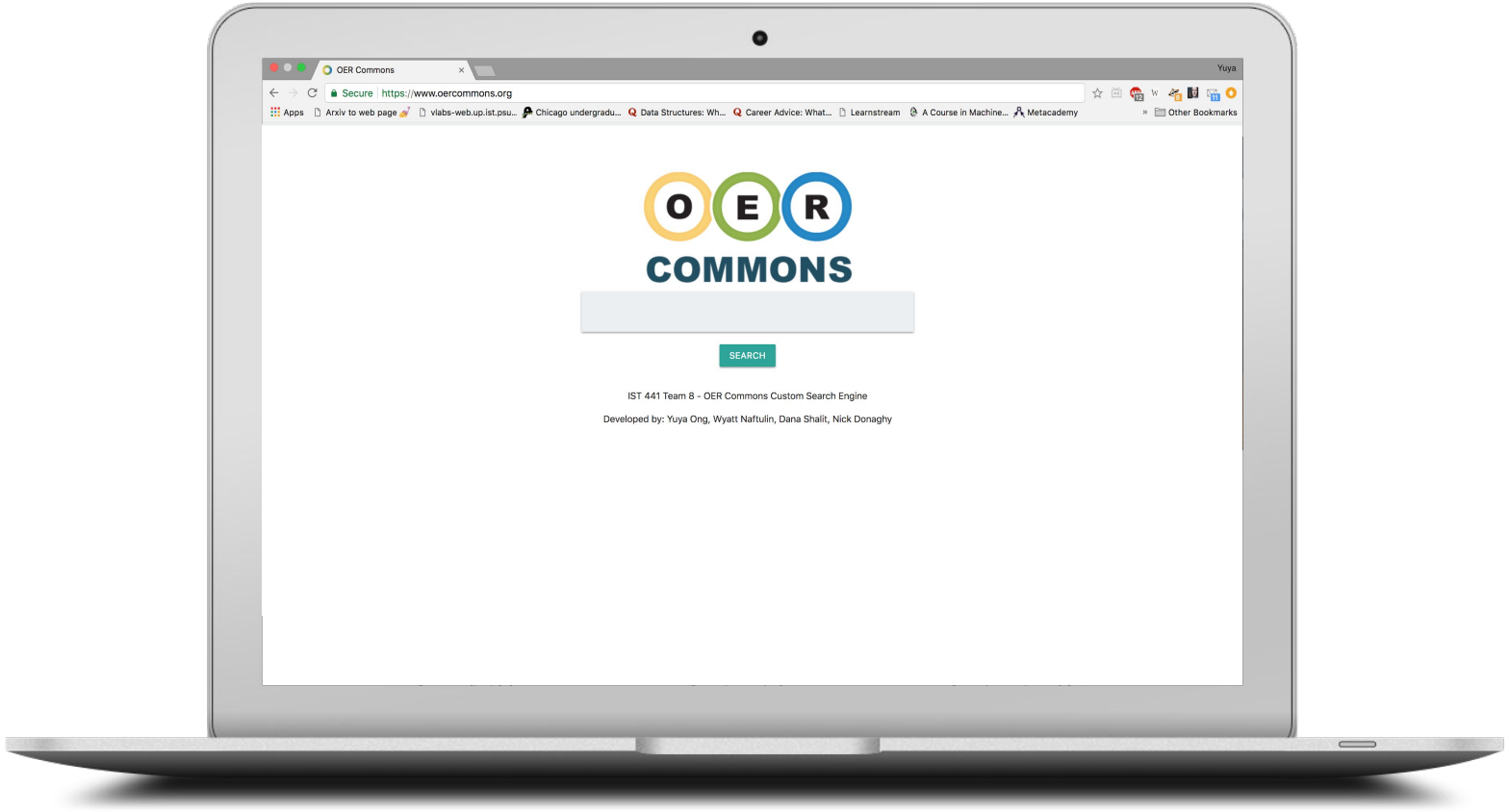


Verb tense



Country-Capital

Front-End Interface



Live Demonstration

http://17ede145.ngrok.io

Empirical Observations & Improvements

- Performs fairly well on general subject terms (i.e. biology, physics, math, etc…)
- Only is able to return relevant results based on what OER Commons has, and nothing else.
- Lots of pages have “Page Has Moved”, 404 Redirects - need to better improve our crawler settings to filter out pages that do not lead anywhere.
- Better error handling should be made for weird queries that come into our system.

Comments from Customer

Further comments from our customer will be provided in the final report.

This will include customer remarks on the overall evaluation of the search engine and the “relative” performance to his expectations.

Conclusion

In this project we have presented:

- Our motivation for building a specialized search engine for OER Commons.
- System architecture and design choices for the Search Engine
- The data processing pipeline and corresponding components.
- Live demonstration of our search engine.

Source Code

The source code for the entire pipeline is available on our Github repository.

https://github.com/yutarochan/IST441