

# CSE 4/535-Introduction to Information Retrieval - Fall 2023

## Project 1 Requirements

Due Date: 27th September 2023, 11:59 PM EST

---

### Introduction

This project aims to introduce students to data scraping and indexing in SOLR. Before mining and analyzing such content, collecting, cleaning, and storing such data is a crucial step. This project aims to familiarize students with collecting online data and efficiently storing them, making data retrieval and analytics easier for downstream applications. This project will also introduce students to the different technical aspects involved in this course and subsequent projects.

**[Caution]:** Make sure you SAVE all the data you have collected/ indexed in a secure location; you may need this in your final project.

### Part 0: Setup

GCP: For this project(and the forthcoming projects), we will use Google Cloud Platform(GCP) as our default cloud platform. Requirements to get a GCP account with \$300 credits:

- A valid Gmail account. Please DO NOT use your buffalo.edu email address.
- A credit/debit card.

**[Caution]:** Make sure you are not crossing your \$300 limit. Otherwise, you may be charged. Please do not keep your instances running unless specifically asked to do so.

How to open a GCP account?

<https://www.youtube.com/watch?v=W5mPX1-015o> or watch any other youtube videos.

How to create a VM instance?

<https://cloud.google.com/compute/docs/instances/create-start-instance>

[nce](https://cloud.google.com/compute/docs/instances/create-start-instance)

<https://www.youtube.com/watch?v=goIKNBMqQhE> or watch any other youtube videos.

How to open a port?

<https://www.youtube.com/watch?v=-RjDWwTZUnc> or watch any other youtube videos.

Apache Solr:

Installation:

<https://tecadmin.net/install-apache-solr-on-ubuntu/>

NOTE: In the above article the URL to download SOLR installer in 'Step 2' may not work. Please use the following instead:

```
wget https://archive.apache.org/dist/solr/solr/9.0.0/solr-9.0.0-src.tgz
```

If wget is not available in VM, use this to install:

```
sudo apt install wget
```

In GCP VMs, Solr runs in localhost, to run Solr which is accessible to public IP follow this link: <https://tecadmin.net/configuring-apache-solr-to-accessible-on-public-ip/>

Tutorials:

[https://www.youtube.com/watch?v=y2TqbjTclGE&list=PLBrWqg4Ny6vXPalbTc\\_QqPiW1\\_G01AE2a](https://www.youtube.com/watch?v=y2TqbjTclGE&list=PLBrWqg4Ny6vXPalbTc_QqPiW1_G01AE2a)

## Part 1: Scraping Wiki pages

*Understanding the Data:*

The screenshot shows the Wikipedia page for 'Information retrieval'. A red box highlights the URL in the browser's address bar: `en.wikipedia.org/wiki/information_retrieval`. Another red box highlights the title 'Information retrieval'. A third red box highlights the summary text, which is also labeled 'SUMMARY' in red. The summary text reads: 'Information retrieval (IR) in computing and information science is the process of obtaining information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full-text or other content-based indexing. Information retrieval is the science<sup>[1]</sup> of searching for information in a document, searching for documents themselves, and also searching for the metadata that describes data, and for databases of texts, images or sounds. Automated information retrieval systems are used to reduce what has been called information overload. An IR system is a software system that provides access to books, journals and other documents; it also stores and manages those documents. Web search engines are the most visible IR applications.'

The page also includes a table of contents on the left, a 'Read Edit View history Tools' bar, and a sidebar on the right titled 'Information science' with links to various related fields and subfields.

We can see a wiki page about 'Information Retrieval'. You will use an API to retrieve the information as given in the red box.

### *Scraping Requirements:*

- Scrape data from Wiki related to the following Topics. Some subtopics are provided as suggestions.
  - a. **Health:** Common diseases, global health statistics, mental health trends...
  - b. **Environment:** Global warming, endangered species, deforestation rates...
  - c. **Technology:** Emerging technologies, AI advancements...
  - d. **Economy:** Stock market performance, job markets, cryptocurrency trends...
  - e. **Entertainment:** Music industry, popular cultural events, streaming platforms...
  - f. **Sports:** Major sporting events, sports analytics...
  - g. **Politics:** Elections, public policy analysis, international relations...
  - h. **Education:** Literacy rates, online education trends, student loan data...
  - i. **Travel:** Top tourist destinations, airline industry data, travel trends...
  - j. **Food:** Crop yield statistics, global hunger and food security...
- Get a minimum of 500 documents for each of these topics listed above
- Make sure all the 500 documents from each topic are unique
- Not more than 5% of documents of length less than 200 characters in the summary

### *API usage:*

**Endpoint:** [https://www.mediawiki.org/wiki/API:Main\\_page](https://www.mediawiki.org/wiki/API:Main_page)

Suggested Wrapper: <https://pypi.org/project/wikipedia/>

[Colab Notebook](#) - Wiki API usage

## **Part 2: Indexing**

Before we describe the indexing process, we introduce some terminologies.

### *Solr Terminologies*

- Solr indexes every document subject to an underlying schema.
- A schema, much akin to a database schema, defines how a document must be interpreted.
- Every document is just a collection of fields.
- Each field has an assigned primitive (data) type int, long, String, etc.
- Every field undergoes one of three possible operations: analysis, index, or query.
- The analysis defines how the field is broken down into tokens, which tokens are retained and which ones are dropped, how tokens are transformed, etc.
- Both indexing and querying at a low level are determined by how the field is analyzed.

Thus, the crucial element is configuring the schema to correctly index the collected docs per the project requirements. Every field is mapped to a type, and each type is bound to a specific tokenizer, analyzer, and filters. The *schema.xml* is responsible for defining the full schema, including all fields, and their types, and analyzing indexing directives.

Although a full description of each analyzer, tokenizer, and filter is out of the scope of this document, a great starting point is at the following page: <https://cwiki.apache.org/confluence/display/SOLR/AnalyzersTokenizersTokenFilters> where you can find tips and tricks for all important elements you might want to use for this project. You are encouraged to start either in a schemaless mode or with the default schema, experiment with different filters, and work your way from there.

### *Preprocessing strategies*

This is where students need to figure out the appropriate way to preprocess their collected data. Overall, there are two overarching strategies that you must consider:

- Using out-of-the-box components and configuring them correctly. For example, the StopFilter can filter out stopwords as specified by a file listed in the schema. Thus, at the very minimum, you would be required to find language-specific stopwords lists and configure the filters for corresponding type fields to omit these stopwords.
- Preprocessing data before indexing to extract the needed fields. For example, you could preprocess the pages to introduce new fields in the JSON response as per project requirements. Here again, it is left to your choice of programming language and/or libraries to perform this task. Solr supports a variety of data formats for importing data (XML, JSON, CSV, etc.). You would thus need to transform your queried data into one of the supported formats and POST this data to Solr to index.

### *Solr Schema API and PySolr*

We will use Solr programmatically using Solr Schema API and PySolr, some

documentation: [https://solr.apache.org/guide/6\\_6/schema-api.html](https://solr.apache.org/guide/6_6/schema-api.html)

<https://github.com/django-haystack/pysolr>

More details will be provided in the class (September 11).

### *Indexing Requirements*

Create a core named **'IRF23P1'**. We will use the same core to index the documents.

Field name	Description	Field Type	Is Custom?
revision_id	Revision Id of the page	string	
title	Title of the page	string	

summary	Summary of the page	text_en	
url	URL of the page	string	
topic	Topic the page is related to	string	Yes

[Colab Notebook](#) - SOLR Indexing Demo

## Submission

You are required to submit a .json file containing the IP address of your GCP instance, port in which Solr is running and the core name. The file should be named [\[ubit\]\\_p1.json](#) (example: spal5\_p1.json)

Example file contents:

```
{
  "ip": "60.42.48.55",
  "port": "8983",
  "core": "IRF23P1",
  "ubit": "spal5"
}
```

**[Caution]**: Please check the validity of the JSON file(Tool to check validity: <https://jsonlint.com/> ) before you submit. If you submit an invalid JSON file, a ZERO score is guaranteed (We mean it).

### *Steps for submission:*

You should upload everything in Brightspace. You need to create a zip file containing the following

1. The JSON file (described above)
2. The codebase in a directory named 'src'.
  - a. It should contain the code used for scraping, pre-processing, and indexing the data in SOLR.
  - b. The data which was indexed in any format.

## Grading

This project is worth **10 points**, and these are distributed as follows:

Criterion	Description	Points
Docs volume	Get a minimum of 5000 docs	3
Min docs per topic	Get a minimum of 500 docs	2
Summary length	Not more than 5% of documents of length less than 200 characters in the summary	2
Preprocessing	Not more than 5% of documents should have any other characters other than alphanumeric.	1
Solr schema validation	All fields are named as required, contain values as required, etc.	2

## Academic Integrity

Academic integrity policies will be taken very seriously. We will check the pattern of your indexed data, and if similarities are found more than a threshold value(which will not be disclosed) with other submissions, you will get a **ZERO** score. If you were found continuously arguing with us regarding your grades(unless there is a fault on our side), your grade would be reduced by up to 50% of your original score, and you would be reported to the department.