WIKI BY PLACE

Extracting Wikipedia entries in a specific area

Introduction

This project was done in the TDK - Technion Data & Knowledge Lab of the CS faculty.

By Nerya Hadad under the supervision of Dr. Oren Mishali.

This presentation purpose is to described and explain the implementation of the project.

In order to understand the project purpose, it’s will be useful to read the README file – until “Getting Started”.

Table of contents

Data resources

- dbpedia-hebrew
- MediaWiki API

High level implementation

- elastic_builder.py
- search.py
- server.py

Data resources

1. **dbpedia-hebrew.**

This is a former project in the TDK.

URL :https://github.com/TechnionTDK/dbpedia-Hebrew

This former project is used to extract 2 json files out of Wikipedia dump files, containing all entries in Wikipedia Hebrew with the following data:

"label" - headline of the Wikipedia page.

"url" - url of the Wikipedia page.

"abstract" - this name is taken from Wikipedia dump files and contain the same data. This data is based on the information in the Wikipedia page. Some labels have no abstract.

As a preparation to my project, I fixed the abstract this project was generating. Then I extract the 2 files to directory named "input".

Data resources

2. **MediaWiki API**

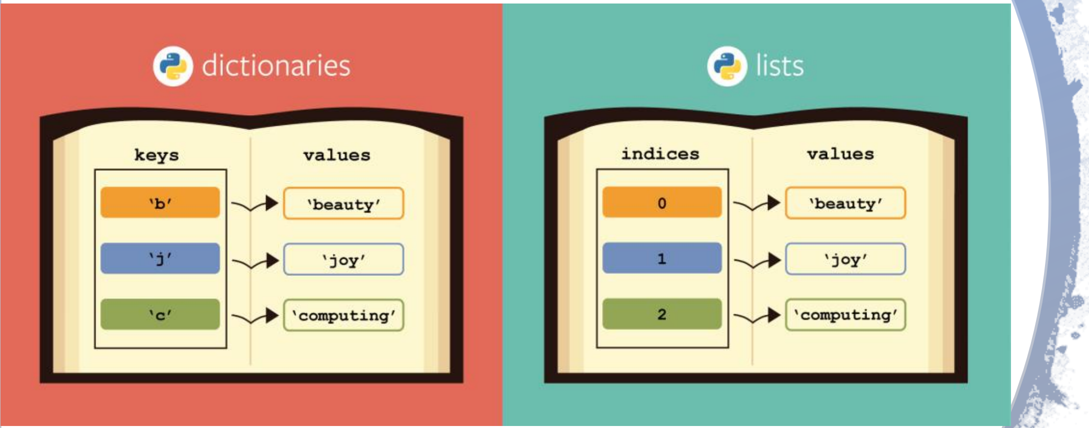
This is an API by Wikipedia.

URL: https://he.wikipedia.org/w/api.php

This API has a lot of uses, the following one is used in this project:

```
For a given Wikipedia page label:
        If the page has no location:
                return the page has no location.
        Else:
                return the location.
```

Notice: not all pages on Wikipedia has location.



Elastic_builder.py

Class dataLoad:

This class is extracting the data from the 2 files in "input" directory (slide 4).

The data is saved in dataLoad's members as public data.

allLabels = [] (list)
labelsToUrls = {} (dict)
urlsToAbstract = {} (dict)

The data is saved this way for complexity reasons of the full script:

The relevant labels would be united after finding their coordinates.

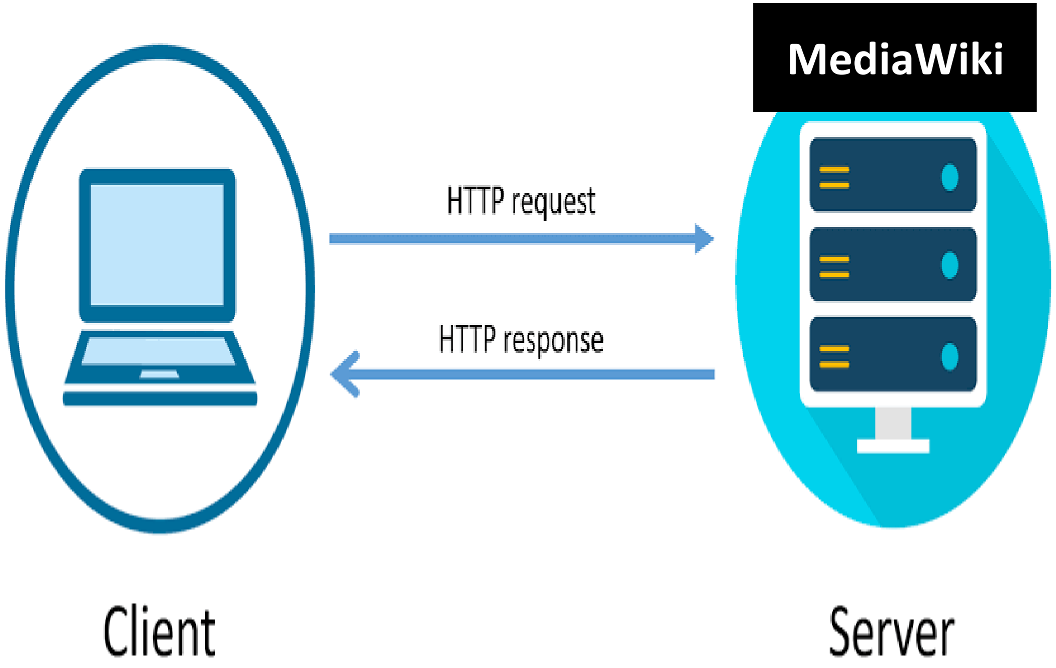
Elastic_builder.py

saveWithCoordinates - method.

Input: one parameter from type dataLoad.

Algorithm:

```
Result = []
For every label at allLabels :
    If label has coordinates (using MediaWiki API):
        Result.append(doc{label})
Return Result
```



doc{label}:
Label
Coordinates
url
Abstract

Elastic_builder.py

elasticBuilder - method.

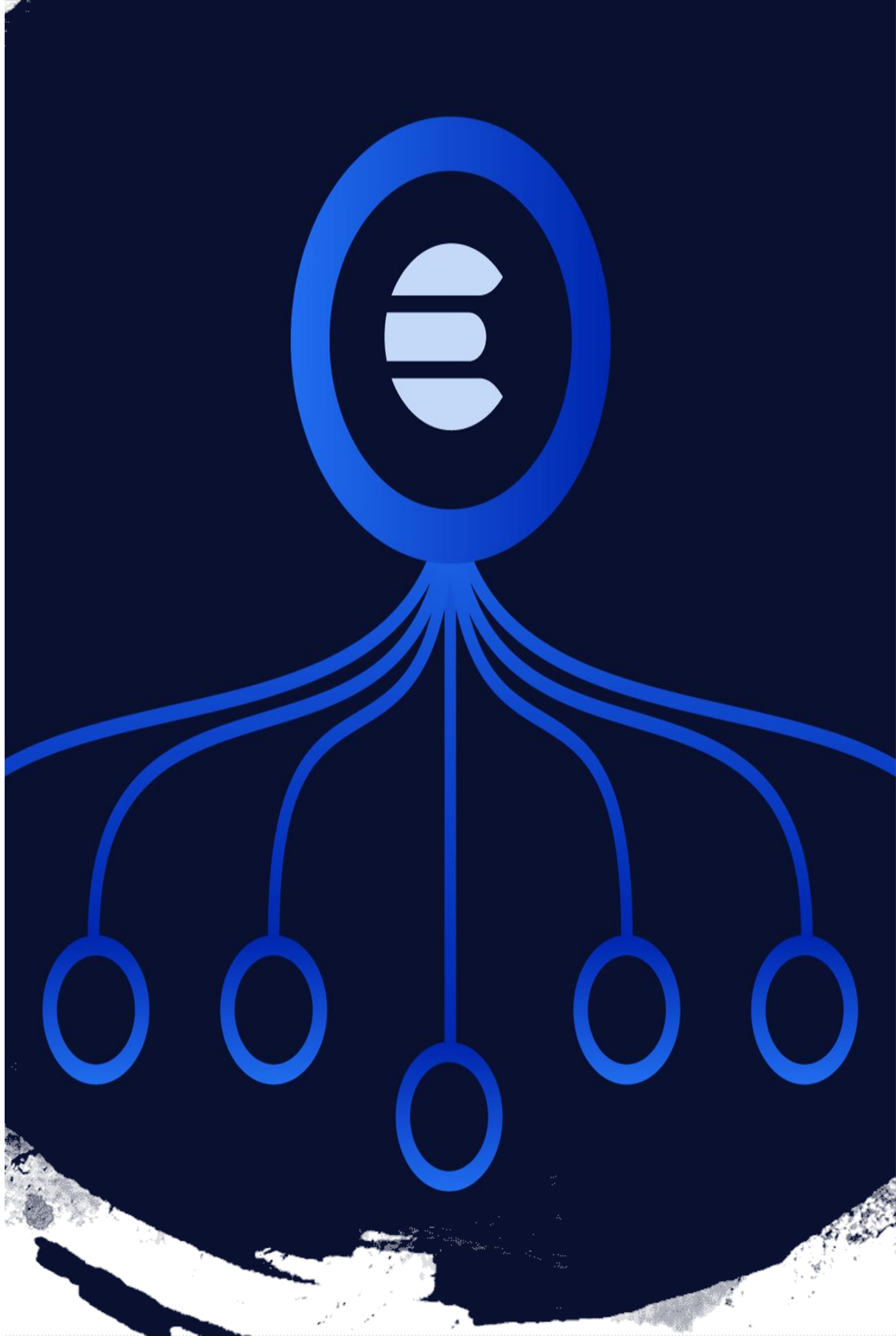
Input: one parameter from type list. Assuming list’s items has coordinates and coordinates are saved in a specific way.

Algorithm:

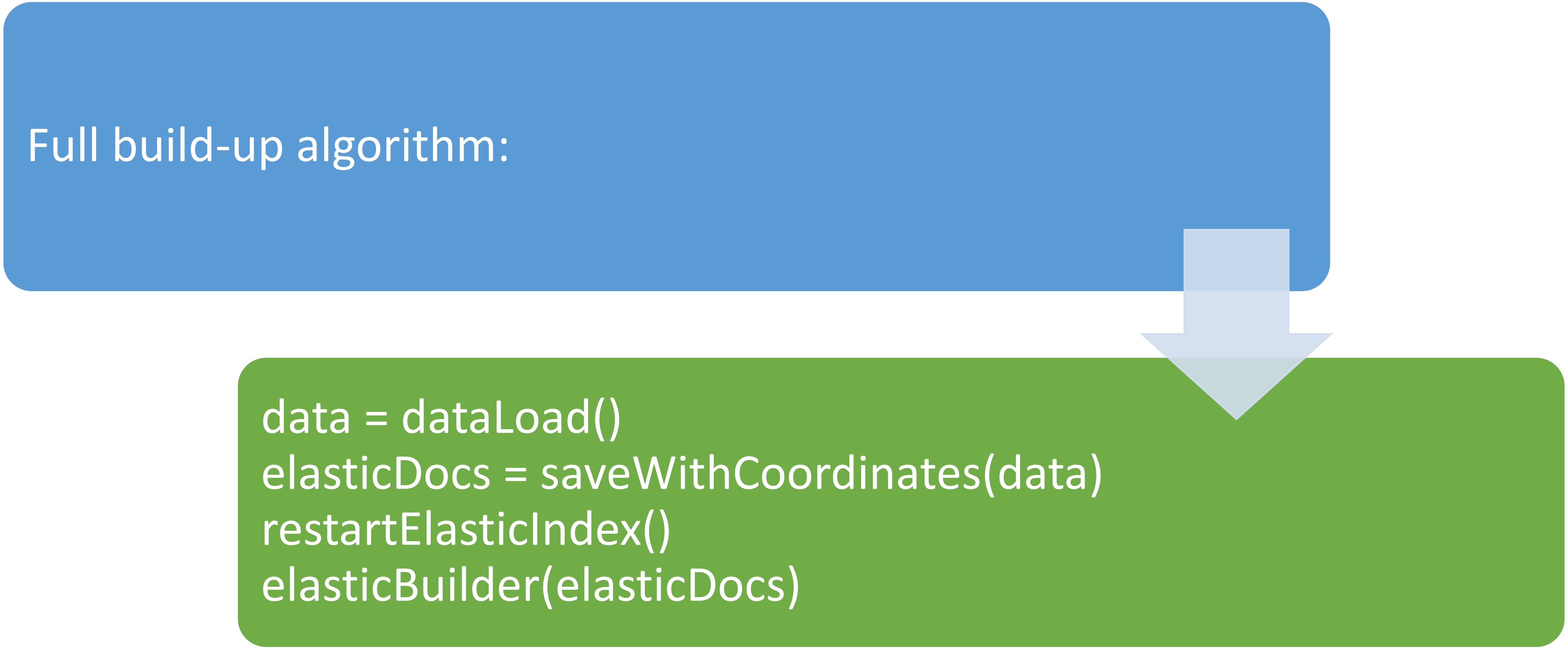
Init elasticsearch engine.

Insert all list’s items to the engine.

restartElasticIndex() is another method that is used to delete the engine in case it’s not the first build.

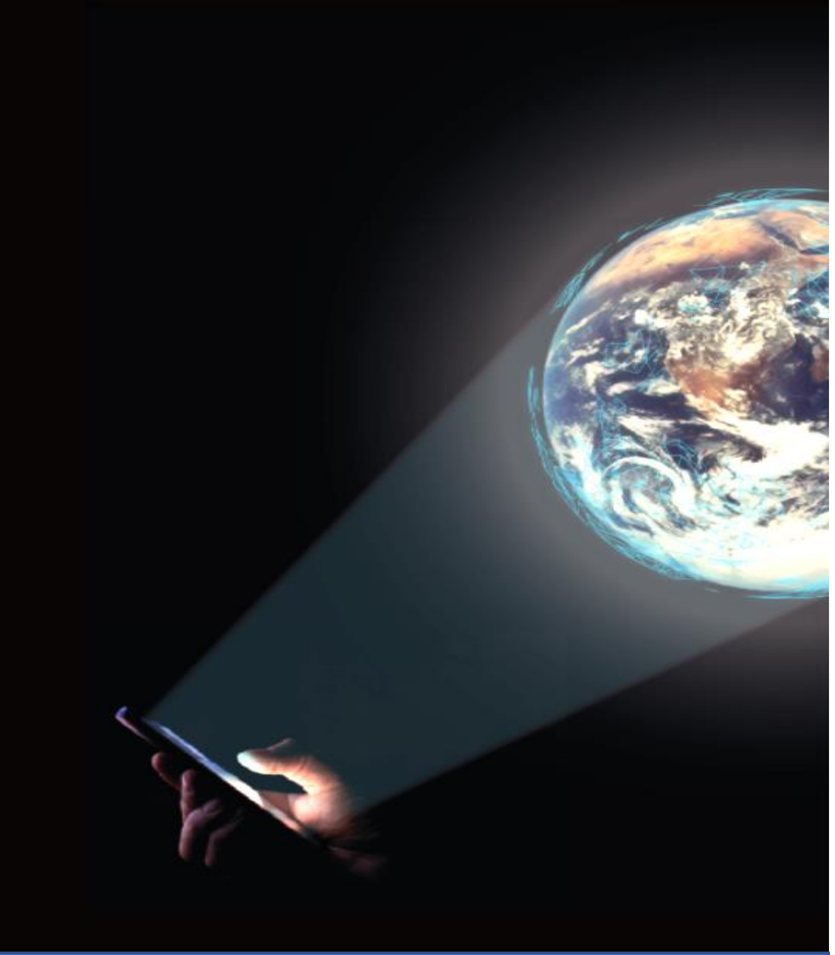


Elastic_builder.py



Full build-up algorithm:

```
data = dataLoad()
elasticDocs = saveWithCoordinates(data)
restartElasticIndex()
elasticBuilder(elasticDocs)
```



search.py

This file is used for the implementation of one method:

search - method.

Input: one parameter from type list.

Assuming list’s items are the following strings:

1. radius - number and its unit [mm, cm, m, km...].

2. Latitude – number. 3. Longitude – number.

Algorithm:

Call the built engine to get doc{label} of all labels in the defined area.

For every label – add the distance from the input.

Return all this data as list of dicts.

doc{label}:

Label

Coordinates

url

Abstract

server.py

This file is a wrapper between the user and the method ‘search’.

It’s using flask to create http request.

Its input is:

location - latitude and longitude ("lat,lon").

radius - number and its unit [mm, cm, m, km…] ("radius").

Algorithm:

Extract the input to the format search is expecting (detailed in the previous slide).

Call ‘search’.

Return ‘search’ result as a json.