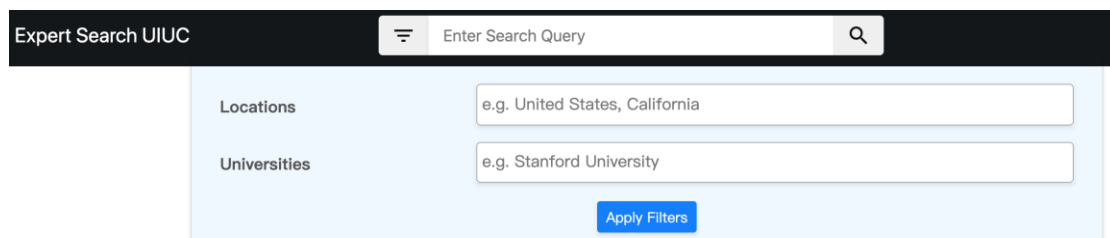


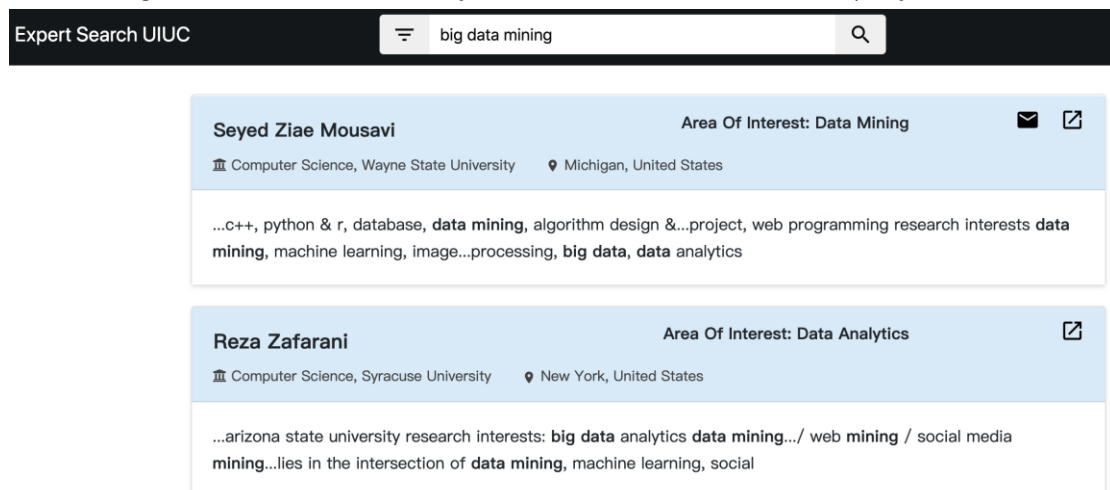
1 Overview of the function of the code

The ExpertSearch system is a webpage-based search engine that provides search function for faculty information. Users can provide a piece of search query that contains name, university, specialization, location, etc. in the text box showing “Enter Search Query”, and hit “Enter” on keyboard or click the icon on the right side of the text box. Two additional filters “Locations” and “Universities” are optional to further filter the desired faculty’s geographic information and association name.



The screenshot shows the 'Expert Search UIUC' interface. At the top, there is a search bar with the placeholder text 'Enter Search Query' and a magnifying glass icon. Below the search bar, there are two filter sections: 'Locations' with a text input field containing 'e.g. United States, California' and 'Universities' with a text input field containing 'e.g. Stanford University'. An 'Apply Filters' button is located at the bottom right of the filter section.

The search result of faculties will display in the order of relevance below. There are two rows in the header area. In the first row, faculty name will show on the left bald-faced, and **Area of interest (the new feature added in this project)** aligns to the right. There can also be a envelop shape icon if email address is captured, and clicking it will evoke the system-default email application to draft an email to the faculty. The next icon will lead to the faculty bio page. In the second row, department and university name display on the left, while the state and country name show on the right. In the main body of search result, it shows a segment from detailed faculty introduction that contains the query information.



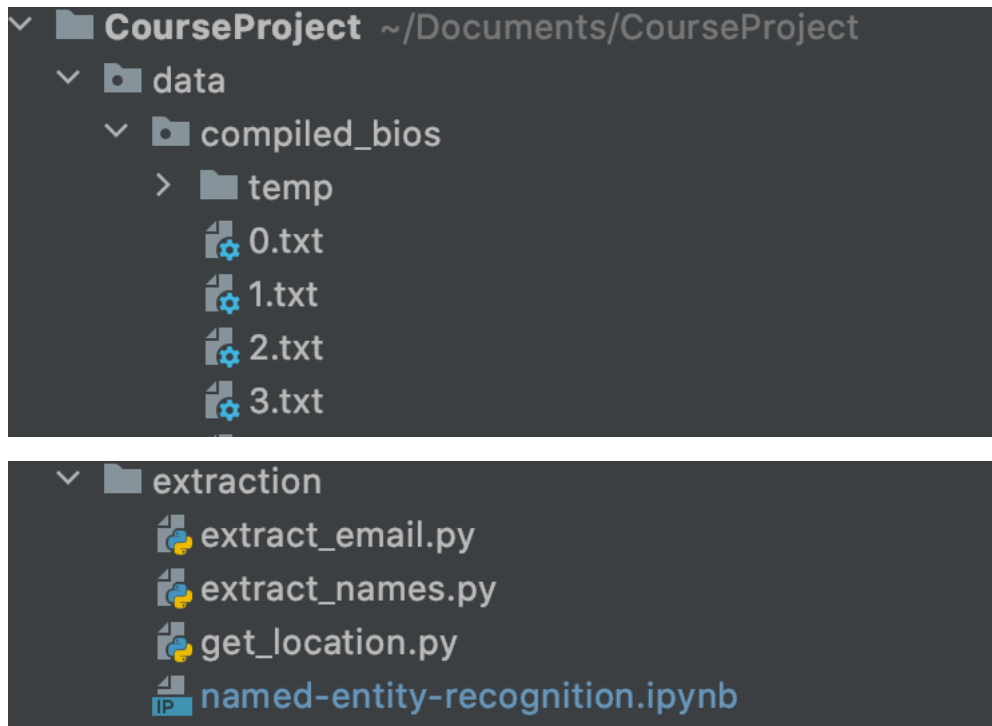
The screenshot shows the search results for the query 'big data mining'. The results are displayed in a list format. Each result entry includes the faculty name, their area of interest, department, university, and location. The first entry is for Seyed Ziae Mousavi, whose area of interest is Data Mining. The second entry is for Reza Zafarani, whose area of interest is Data Analytics. Each entry also includes a brief snippet of the faculty's introduction text.

Faculty Name	Area Of Interest	Department	University	Location
Seyed Ziae Mousavi	Data Mining	Computer Science	Wayne State University	Michigan, United States
Reza Zafarani	Data Analytics	Computer Science	Syracuse University	New York, United States

2 Code structure and implementation details

2.1 Information extraction

The first step of the code is to extract information from the source data. The raw data is saved in the `./data/compiled_bios` folder. Each file is named by a 0-based index and contains one line of faculty detailed introduction. The existing code has 3 .py files in the `./extraction` folder, each has the Python code to extract email, names and location from the raw data.



This project added a Jupyter notebook called “*named-entity-recognition.ipynb*” that can extract the research area of interest from the raw data. Below is a brief introduction of the code logic:



- **Data Preparation**

The dataset we use is the ‘faculty dataset’ collected from CS 410 class. While there are 6525 numbers of documents in the dataset, we randomly sampled 25 documents for training. Then, we tokenized sentences and labeled target tokens into corresponding entities. For example, we labeled ‘database systems’ as a ‘research area’ in the following sentence:

“His primary research interests are database systems.”
("His primary research interests are database systems, object-oriented systems and software engineering.", { 'entities': [(35, 51, 'AREA'), (53,76, 'AREA'), (81, 101, 'AREA'),] })

- **Language Modelling**

While Spacy provides a pre-trained CNN model, we decided to build a new model from scratch with 70 numbers error-free entities. We used 0.5 for the drop rate and default optimizer provided in Spacy.

- **Rule based Matching**

To maximize the quality of outcome, we also leverage rule-based matcher using regular expressions. Since the entities we are looking for are limited to ‘research interest’, we first searched a sentence that includes ‘research’ or ‘interest’ in the validation dataset. We

observed that the entities we aim to extract have a simple structure. For example, they are "My research interest is ..., Research interests are."

● Testing

We obtained qualitative results when performing interpolation in our initial model. We will continue testing our model on the testing dataset for future work. Here are a few examples of research area extracted:

Entities [('Distributed computing', 'AREA'), ('Analysis of algorithms', 'AREA'), ('Data structures', 'AREA'), ('Computational geometry', 'AREA'), ('Graph algorithms', 'AREA')]
Entities [('computer networking', 'AREA'), ('computer security', 'AREA')] *Entities [('CS Education', 'AREA')]*

Extracted information are saved under `./data` folder, each named by the field extracted. For example, in the `names.txt` file, there are 6525 rows. Each row corresponds to the faculty name extracted from the raw data at the same index.

CourseProject ~/Documents/CourseProject	1	Tarek
data	2	Sarita V. Adve Richard T. Cheng
compiled_bios	3	Donald B. Gillies
expertsearch	4	Gul
FacultyDataset	5	
filter_data	6	Bjarne Stroustrup
__init__.py	7	Lawrence Christopher Angrave
areas	8	Brian P Bailey
depts	9	Adam Bates
emails	10	Mattox Beckman
location	11	Marco Caccamo
MP2_Part1 Signup - Sheet1.csv	12	Matthew Caesar
names.txt	13	Roy H Campbell Sohaib
unis	14	Geoffrey Werner Challen
urls		

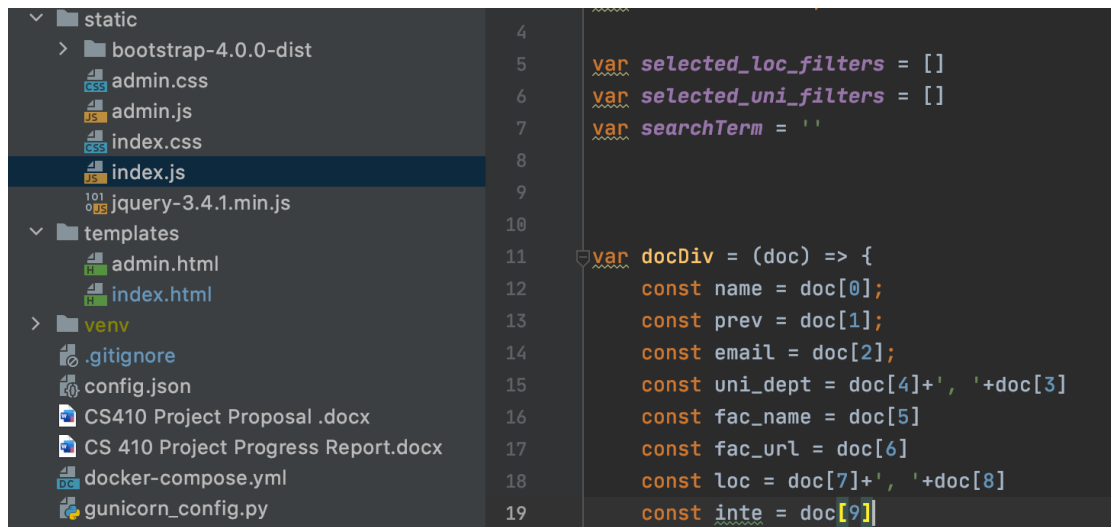
2.2 Ranking function

Those extracted information from last step are merged into a `metadata.dat` file under `./data/compiled_bios` folder. The existing code already has 8 fields in each row in the metadata, so this project created a `./data/compiled_bios/merge_new_field_to_metadata.py` file to append the newly extracted research area to the end of each row. In addition, it's also required to add the name and type to the end of `file.toml` file under the same folder, so that the code knows how to parse the new field added. The metadata file is then feed into the ranker located as `./data/expertsearch/ranker.py`, configured by a config file located at `./data/compiled_bios/config.toml`.

6522.txt	1	<code>type = "file-corpus"</code>
6523.txt	2	<code>list = "dataset"</code>
6524.txt	3	<code>metadata = [{name = "doc_name", type = "string"},</code>
__init__.py	4	<code>{name = "university", type = "string"},</code>
config.toml	5	<code>{name = "department", type = "string"},</code>
dataset-full-corpus.txt	6	<code>{name = "fac_name", type = "string"},</code>
file.toml	7	<code>{name = "fac_url", type = "string"},</code>
merge_new_field_to_metadata.py	8	<code>{name = "state", type = "string"},</code>
metadata.dat	9	<code>{name = "country", type = "string"},</code>
expertsearch	10	<code>{name = "email", type = "string"},</code>
__init__.py	11	<code>{name = "interest", type = "string"}]</code>
ranker.py		

2.3 Visualization webpage

Visualization is handled by the `./templates/index.html` file, which designs the layout and format of the ExpertSearch webpage. It's also controlled by `./static/index.js` file to interact with the ranker and user. The new field "Area of Interest" needs to be added in the JavaScript file, in order to display on the search result.



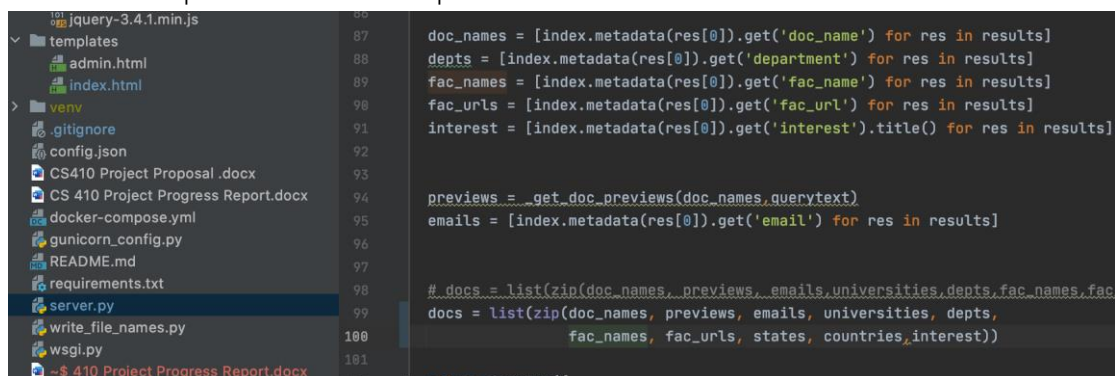
```

4
5   var selected_loc_filters = []
6   var selected_uni_filters = []
7   var searchTerm = ''
8
9
10
11   var docDiv = (doc) => {
12     const name = doc[0];
13     const prev = doc[1];
14     const email = doc[2];
15     const uni_dept = doc[4]+' '+doc[3]
16     const fac_name = doc[5]
17     const fac_url = doc[6]
18     const loc = doc[7]+' '+doc[8]
19     const inte = doc[9]]

```

2.4 Main function

The main function is called `./server.py`, that is executed when the program launches. It uses flask to drive the ranker and visualization functions, so that a local server is running at <http://localhost:8095/>. It's also necessary tell the code to get the research interest in metadata and pass it to the Javascript.



```

87 doc_names = [index.metadata(res[0]).get('doc_name') for res in results]
88 depts = [index.metadata(res[0]).get('department') for res in results]
89 fac_names = [index.metadata(res[0]).get('fac_name') for res in results]
90 fac_urls = [index.metadata(res[0]).get('fac_url') for res in results]
91 interest = [index.metadata(res[0]).get('interest').title() for res in results]
92
93
94 previews = _get_doc_previews(doc_names,querytext)
95 emails = [index.metadata(res[0]).get('email') for res in results]
96
97
98 # docs = list(zip(doc_names, previews, emails, universities, depts, fac_names, fac
99 docs = list(zip(doc_names, previews, emails, universities, depts,
100               fac_names, fac_urls, states, countries, interest))
101
102 return jsonify({

```

3 Installation and usage

It's extremely easy to install and run the code on Mac or Linux, but needs some tricky settings to execute on Windows. This is because the Python webserver package Gunicorn doesn't support directly on Windows system, so Docker is required.

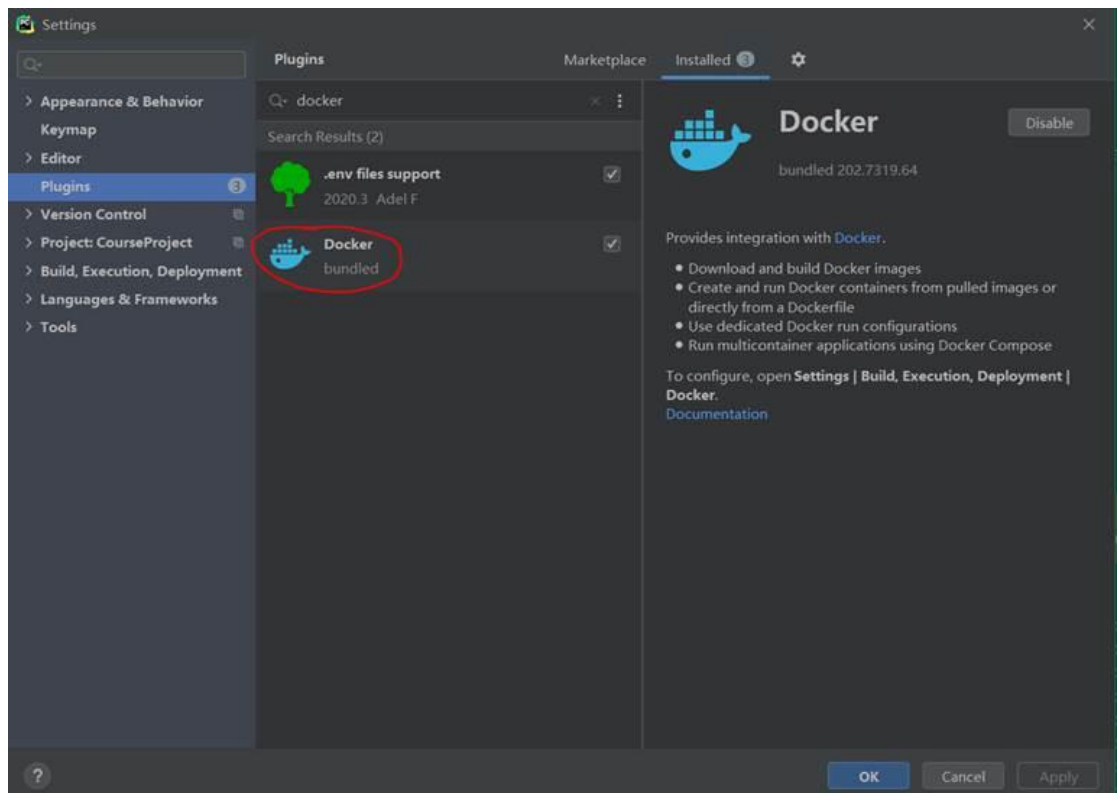
3.1 Installation on Windows

- Install Git (if you haven't)
- In the folder you want to save the project, right click -> Git Bash Here -> type: git clone <https://github.com/losaohan/CourseProject.git>
- Download and install Docker Desktop for Windows: <https://hub.docker.com/editions/community/docker-ce-desktop-windows/>
- Download and install Pycharm Professional for Windows: <https://www.jetbrains.com/pycharm/download/#section=windows>

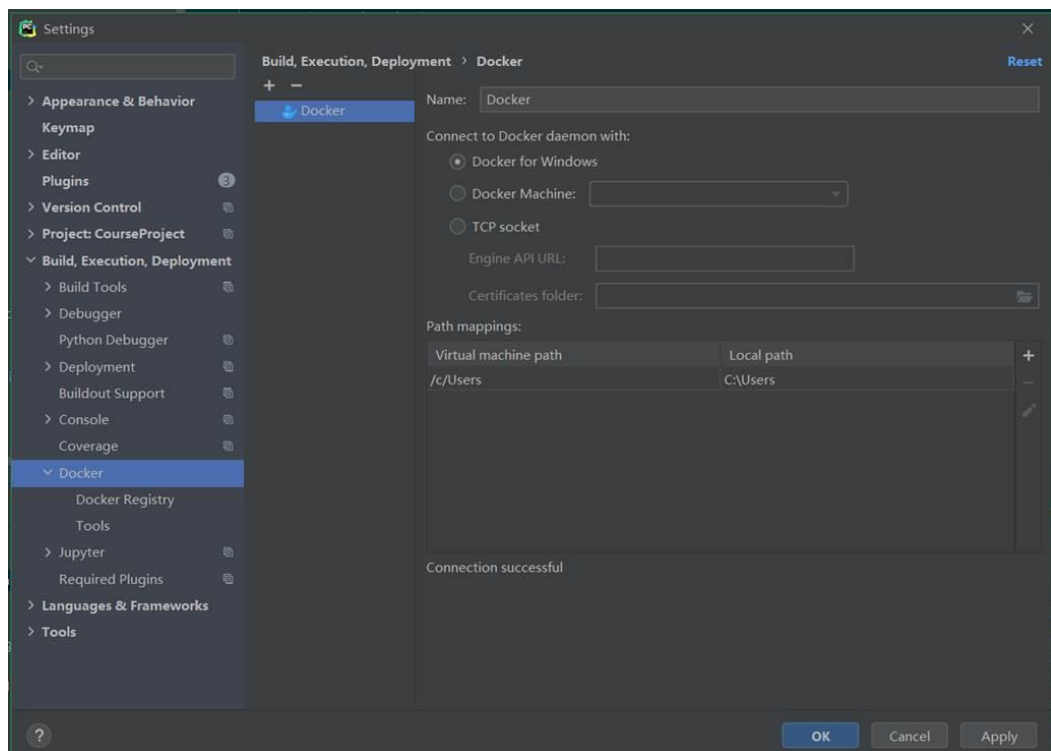
You need to apply an JetBrains account with your Illinois email to have access to

the professional version: <https://www.jetbrains.com/shop/eform/students>.

- Open Pycharm, open the CourseProject folder as project. On the top-left menu, hit File -> Settings -> Plugins -> search for Docker, and install it -> make sure it's "Enabled":

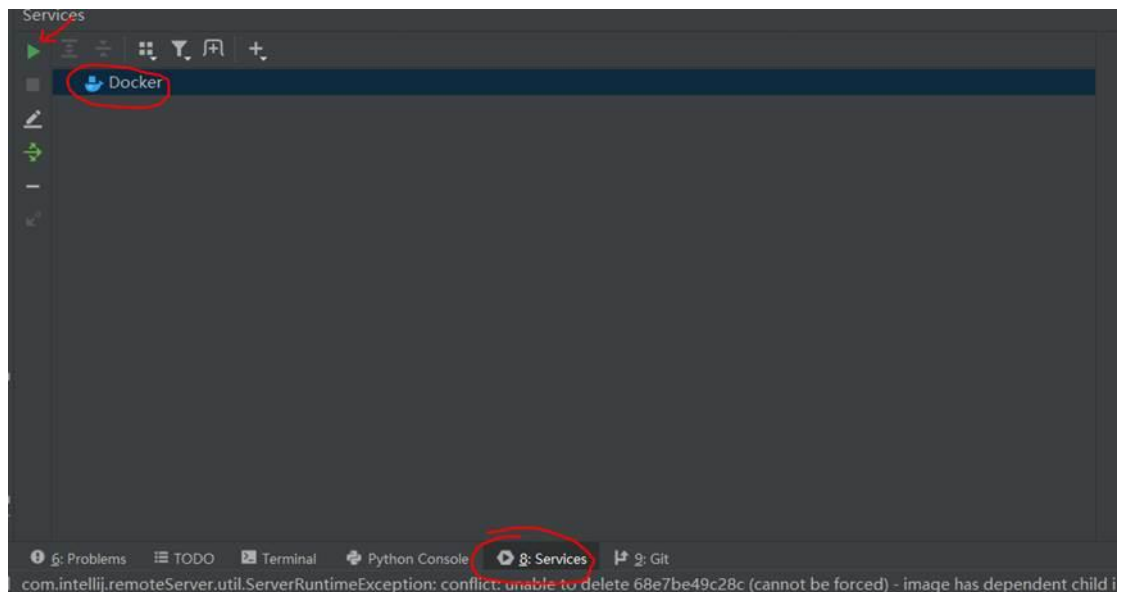


- In the Settings -> Build, Execution, Deployment -> Docker, create a new Docker service as follows -> hit OK.

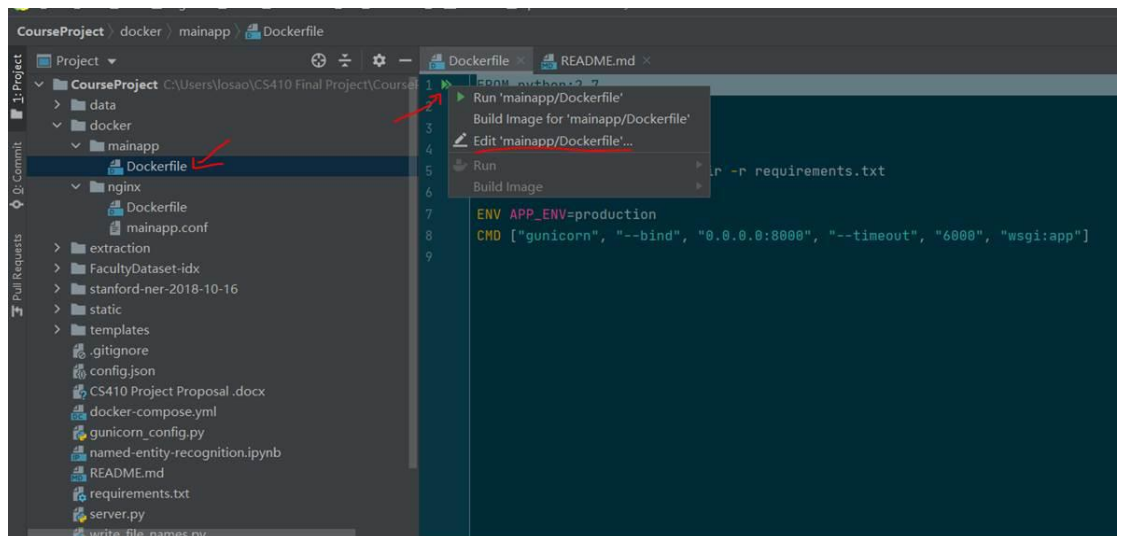


A new Docker service will appear on the bottom left menu. Make sure the Docker

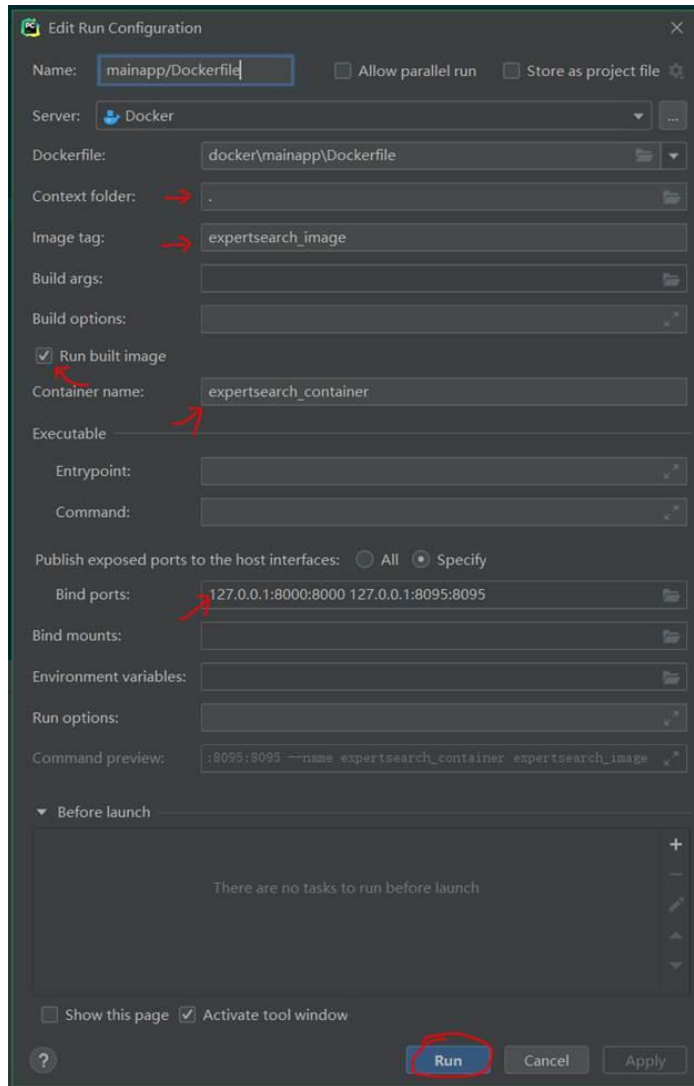
Desktop you installed in step 1 is running, and hit the green button “connect” on the top left: it will show “Connected” in the middle of the bottom window.



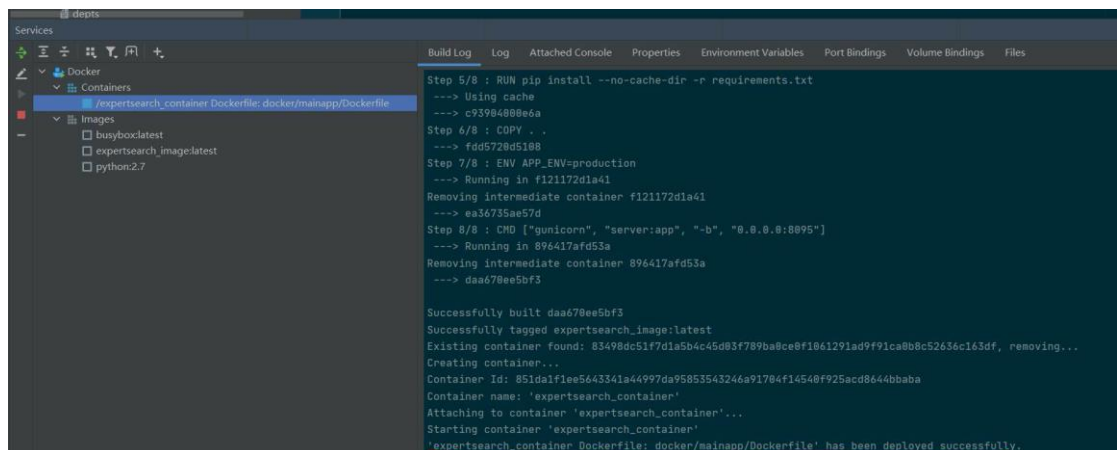
- In the Project directory, open the file docker/mainapp/Dockerfile. Right click the double green arrow and select “Edit ‘mainapp/Dockerfile’...”.



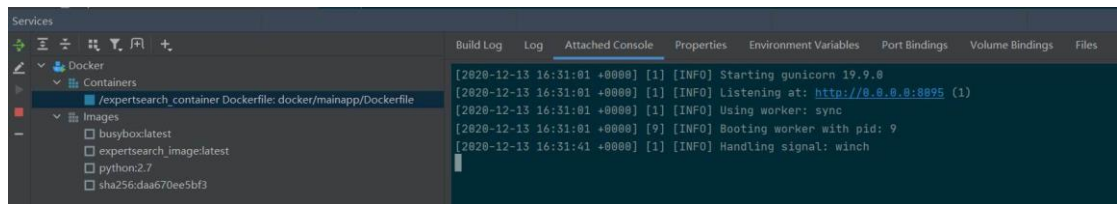
- Fill in the same setting as below in the pop up config edit window. You could make your own Image Tag and Container Name, but make sure the other settings are the same.



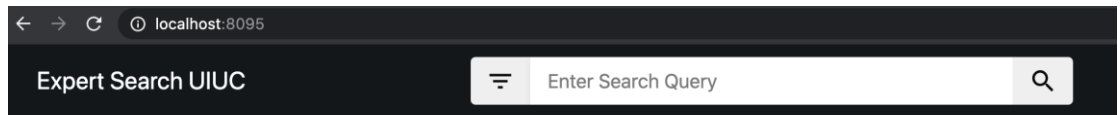
Hit Run: the image will start to build, and the container will launch in a few seconds.



- In the attached console, you will find the code is running at 0.0.0.0:8095 in the container, which is mapped to 127.0.0.1:8095 in the host machine.

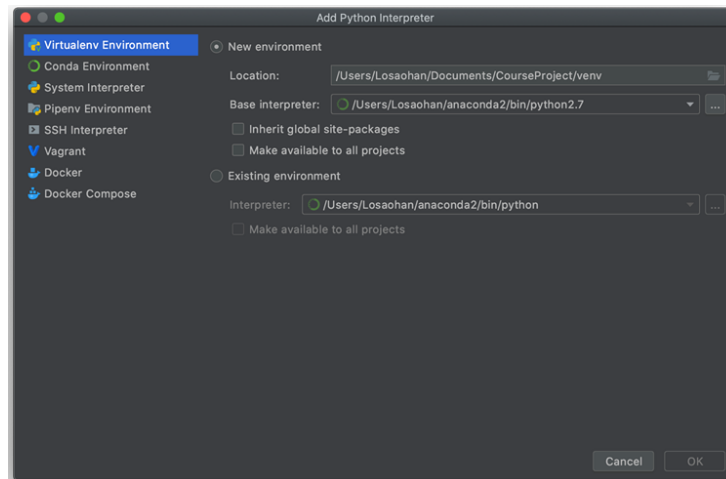


If you type localhost:8095 in your browser, the expert search page will show up

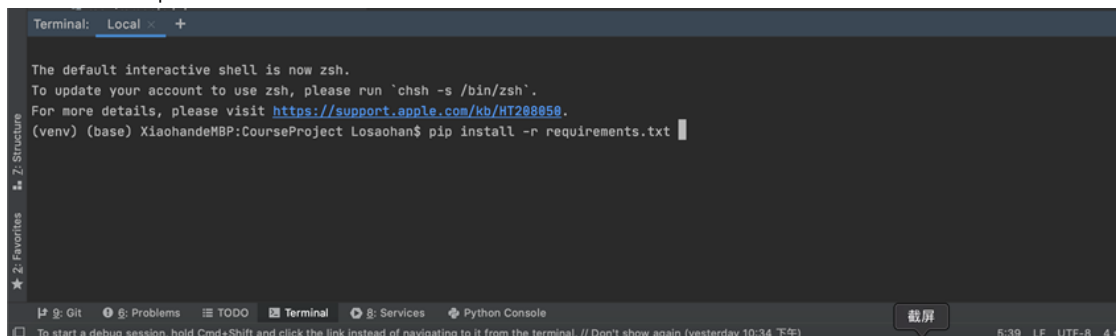


3.2 On Mac/Linux

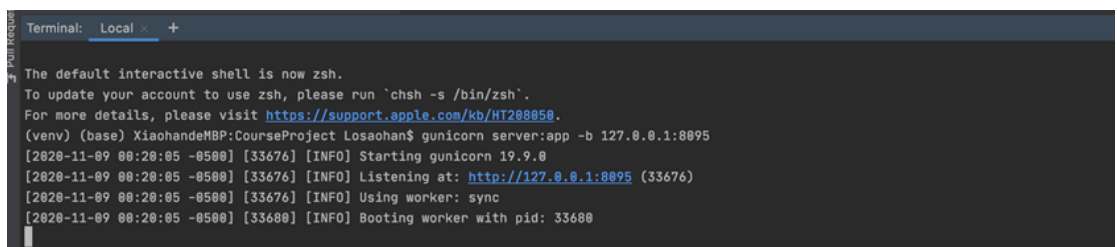
- Open Pycharm, open the CourseProject folder as project. Create a new virtual env by choosing your local Python 2.7 as base interpreter.



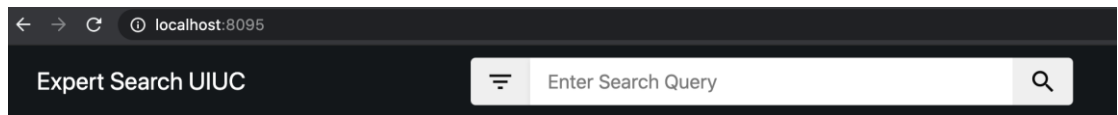
- In the activated virtual env, Install the requirements.txt packages by typing: `pip install -r requirements.txt`



- Type `gunicorn server:app -b 127.0.0.1:8095` in the terminal and the server will launch.



Type localhost:8095 in your browser, the expert search page will show up:



4 Team member contribution

Bruno Seo: Focused on the modeling part: analyzed on the source data, identified the pattern of research interest key words, implemented the algorithms of Named Entity Recognition, and trained the extraction model.

Joseph Angulo: Focused on the front-end visualization part: figured out how the web server work, designed the appearance of updated webpage, and implemented the changes to show the new field on the page.

Xiaohan Liu (captain): Coordinate and focus on the high-level Python: solved the installation and execution across Windows/Mac/Linux systems, figured out the structure and workflow of the project, implemented code to prepare training data as well as code to create metadata, drafted the proposal / progress report / final document / presentation.