

- Software code submission with documentation (65%): Due end of Week 15 (Dec 09, 2021), each project team will be asked to submit the produced source code with reasonable documentation. The documentation should cover both how to use the software and how the software is implemented. The 65% of the grade would be distributed as follows: 45% for source code submission; 20% for documentation submission. Both would be graded based on completion.

CS410 Course Project Summary

Team Name: SearchExperts (Fall 2021)

1. Course Project:

Expert Search

2. Team Git Repository:

<https://github.com/genggeng88/CourseProject>

3. Uploaded Code files:

- a. **scraper.py**
- b. **BinaryClassifier folder:** preProcess.py, BinaryClassifier.ipynb, urls_positive.txt, urls_negative.txt, bio_Binary.csv
- c. **Topic.ipynb**

4. Tutorial Presentation Link:

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

5. Team members:

- a. Qin Geng (genggeng2@illinois.edu) (Captain)
- b. Aditya Vyas (vyas7@illinois.edu)
- c. Junwei Chen (junweic4@illinois.edu)
- d. Peter Zhang (ajun2@illinois.edu)

Abstract:

Searching is a significant part of Text Information Retrieval and building a search engine is more than meaningful to text searching. In our project, we aimed at improving the existing expertise searching engine ([ExpertSearch](#)) from two aspects. First, we will help in automatically scraping faculty links from a random website (Task 2 ~ 4). Second, we will extract more information, including the professor's telephone number and research areas (Task 5 ~ 6). Our work on improving this project is meaningful for both project developers and users. With our efforts on automatically scraping, the project developers can spend less time on manually scraping and get more data meanwhile. As we extract more information, the users of the search engine can get more information about the professors when they are searching.

1. Introduction

The Expert Search was developed by some previous students as part of their course project, based on the dataset of faculty links and bios. The links and bios are scraped from a given department link by each student manually in MP2.2. After we set up the environment for Expert Search, we can do expertise searching in it, as shown in Figure 1.1. This search engine can also apply filters, like locations and universities to narrow down the searching results.

mechanical engineering

Locations

e.g. United States, California

Universities

e.g. Stanford University

Apply Filters

Electrical and Computer Engineering, University of Southern California

California, United States

["part-time lecturer of electrical **engineering** - systemseducation2008, doctoral degree,...electrical **engineering**, university of southern california2003,...master's degree, electrical **engineering**, university of southern california1999,

Cheng Kung

Electrical and Computer Engineering, University of Southern California

California, United States

["part-time lecturer of electrical **engineering** - systemseducation1988, doctoral degree,...electrical **engineering**, university of southern california1979,...master's degree, **mechanical engineering**, san diego state

John

Electrical and Computer Engineering, University of Southern California

California, United States

...shea early career in civil **engineering** and associate professor of...civil and environmental **engineering**, industrial and systems **engineering**,...and electrical **engineering**-systemseducationdoctoral degree, electrical **engineering**, university

Figure 1.1

2. Related Work

Our team's related work contains two parts: automatically scraping links and extracting more information. As we discussed in the Introduction part, the dataset Expert Search uses is scraped from a given department link by each student manually. Our team will help with automatically scraping the urls and bios from a given university website. After doing the searching, we can tell from Figure 1.1 that the shown information is kind of limited. Our team extracted more information, including telephone number and research areas, and showed the extra information on the search page. Table 2.1 shows the content of each specific task. Our team has completed all the tasks listed in our proposal and reached a workload hour of 180. The next part will give more detailed descriptions for each task.

Table 2.1

Task No.	Task Description	Workload	Completion
----------	------------------	----------	------------

3

Task 1	Setting up the environment	35hr	Completed
Task 2	Writing a function of automatically crawling random web pages, and collecting them for the next step use.	10hr	Completed
Task 3	Building a regex-based model to identify faculty directory webpages. Improving this model to achieve an accuracy of 70%.	20hr	Completed
Task 4	Building a binary classifier model to identify bio_url. Improving this model to achieve an accuracy of 70%.	30hr	Completed
Task 5	Writing an extracting function based on Natural Language Processing techniques to recognize and extract names/profiles in web links.	15hr	Completed
Task 6	Writing a topic mining function in extracting common search areas, and displaying it in the web page.	25hr	Completed
Task 7	Integrate all functions and models to the original project (ExpertSearch) and test the whole project.	35hr	Completed
Task 8	Weekly meeting and discussion	10hr	Completed
Total		180hr	





3. Description and Significance

3.1 Setting Up Environment

As the readme in the original project is very simple, and Python 2.7 is outdated, we spent a lot of time to find a solution to set up an environment. We tried Oracle Linux box, Docker, AWS Cloud 9, AWS EC2. At last, At last,

1. we decided to use Windows 10 + Ubuntu 18.04 to set up ENV
2. We changed the server code a little bit to ensure we can really query via Web page;

To share the experience, we created the following readme.

..		
 how_to_setup_env.txt	Create how_to_setup_env.txt	2 minutes ago
 how_to_test.pdf	Create how_to_test.pdf	12 seconds ago
 how_to_test_basic_extractor.docx	Create how_to_test_basic_extractor.docx	38 minutes ago
 how_to_test_basic_extractor.txt	Create how_to_test_basic_extractor.txt	38 minutes ago

For more details, you may go to the following site:

<https://github.com/peterzhangon/Final410/tree/master/readme>

```
88 lines (78 sloc) | 2.14 KB
1 # Part-1: Overview of the development ENV
2 # =====
3 # Basically, we install Ubuntu 18.04 on windows 10 box.
4 # 1: We write code using Python IDE on windows box;
5 # 2: We run the web service on Ubuntu;
6 # 3: We using Google chrome on Windows to query and display;
7
8 # Part-2: To install ubuntu on desktop directly
9 # =====
10 # Go to microsoft store, and search ubuntu 18.04, and install it
11 # https://www.youtube.com/watch?v=X-DHaQLr8i8
12 # You may set up Ubuntu username and password.
13
14 # Part-3: To install packages on Ubuntu
15 # =====
16 #
17 sudo apt-get update
18 sudo apt install python
19 sudo apt install python-pip
20 sudo apt install gunicorn
21 pip install flask
22 pip install metapy
23 pip install requests
24
25 # Part-4: To clone the code via git bash on windows or via terminal on Ubuntu
26 # =====
27 # We may put codes in fixed location, so we can access the code via Ubuntu and Windows
28 #
29
30 # The directory on Ubuntu
31 cd /mnt/c/code
32
33 # The folder on windows
34 cd c:\code
35
36 # Part-5: pip list results
37 # =====
38 # If you cannot start web service and query via web page, you may check the required packages have been installed or not.
39 #
40
41 peter@LAPTOP-MSKOE3G5:/mnt/c/peter/code/Final410$ pip list
```

3.2 Scraping the web-pages

Task 2 primarily focuses on scraping web pages from the internet. To implement it, we created a scraper in the Python programming language using Requests and BeautifulSoup4.

Through this task we were able to fetch details and content of the page for future use.

(You can find the code in scraper.py file as a function names as `get_links`)

This task was tricky as we needed

& 3.3

3.4 Binary Classifier

Task 4 mainly focuses on building and training a binary classifier to predict whether a given link is a faculty link or not. This task contains two smaller parts: data pre-processing and classifier training.

(You can get the complete code from “preProcess.py” and “BinaryClassifier.ipynb”)

3.4.1 Data Pre-Processing

From MP2.1, we already got abundant faculty links, contributed by current and previous students from this course. This part of data, including 16,492 links, can be the positive dataset for the classifier. We also need some negative dataset, namely the links that are identified as non-faculty links. In order to get the negative dataset, we wrote a code snippet to automatically scrape random links from a given list of non-educational websites, as shown in Figure 3.4.1.

```

base_urls = ['https://www.coursera.org', 'https://www.apple.com', 'https://www.linkedin.com',
.....
..... 'https://github.com', 'https://stackoverflow.com', 'https://aclibrary.org',
..... 'https://www.youtube.com', 'https://twitter.com/', 'https://store.google.com',
..... 'http://www.google.com', 'https://www.udemy.com/', 'https://harness.io/',
..... 'https://blockchain.berkeley.edu/', 'https://openclassrooms.com/',
..... 'https://campuswire.com/', 'https://docs.oracle.com/',
..... 'https://algs4.cs.princeton.edu/', 'https://www.outreachy.org/',
..... 'https://world.taobao.com/']

f = open('urls_negative.txt', 'w')
for url in base_urls:
    reqs = requests.get(url)
    soup = BeautifulSoup(reqs.text, 'html.parser')
    s = 'http'
    for link in soup.find_all('a'):
        url2 = str(link.get('href'))
        if s in url2:
            f.write(url2)
            f.write('\n')
        else:
            url_new = url + url2
            f.write(url_new)
            f.write('\n')
f.close()

```

Figure 3.4.1

All the non-faculty links are automatically scraped and saved in the “urls_negative.txt” file.

Next processing is to add attributes to all the data as input to our classifier. As we can see from Figure 3.4.2, there is a collection of keywords. We set these keywords as attributes, and the value of a specific attribute will be marked “1” if the link contains the corresponding keyword and marked “0” inversely. The last label “outcome” denotes the link is a faculty link with marking it “1” and marking it “0” inversely. All the data after processing will be saved in the “bio_Binary.csv” file as input of the classifier.

```

# collection of all the attributes
keywords_ = ['faculty', 'staff', 'people', 'professor', 'bio', 'index', 'id', 'profile', 'outcome']

# Write attribute into the data file
my_data = open('bio_Binary.csv', 'w')
s = ','
keyword_str = s.join(keywords_)
my_data.write(keyword_str)
my_data.write('\n')

m = len(contents_p)
n = len(contents_n)
k = len(keywords_)

my_data_arr = ['']*(m+n)
for i in range(m):
    for j in range(k-1):
        if keywords_[j] in contents_p[i]:
            my_data_arr[i] += ('1,')
        else:
            my_data_arr[i] += ('0,')
    my_data_arr[i] += ('1')
    my_data.write(my_data_arr[i])
    my_data.write('\n')

```

Figure 3.4.2

3.4.2 Classifier Training

After we collected and pre-processed all the data, we can start building and training our classifier. We first randomly shuffled the dataset and divided it into two parts: training dataset and evaluation dataset, and separated each part of data into features and outcomes. We used the scikit learn package in this part and chose the Gaussian Naive-Bayes model as our classifier. After training the Gaussian Naive-Bayes classifier with our training features and training outcome, we predicted if a given link is a faculty link or not.

4 Train Classifier

We first train the classifier gnb with the built-in Gaussian Naive-Bayes model with our train_features and train_labels. Then we did the prediction for both the train features and evaluation features. Finally we got the accuracy of over 70% for both training dataset and evaluation dataset.

```

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(train_features, train_labels)
train_pred_sk = gnb.predict(train_features)
eval_pred_sk = gnb.predict(eval_features)
print(f'The training data accuracy of your trained model is {(train_pred_sk == train_labels).mean()}')
print(f'The evaluation data accuracy of your trained model is {(eval_pred_sk == eval_labels).mean()}')

```

The training data accuracy of your trained model is 0.7343033256880734
The evaluation data accuracy of your trained model is 0.7294353683003726

Figure 3.4.3

Finally, as shown in Figure 3.4.3, we got the prediction accuracy of over 70% for both the training dataset and the evaluation dataset which achieved our goal in our project proposal.

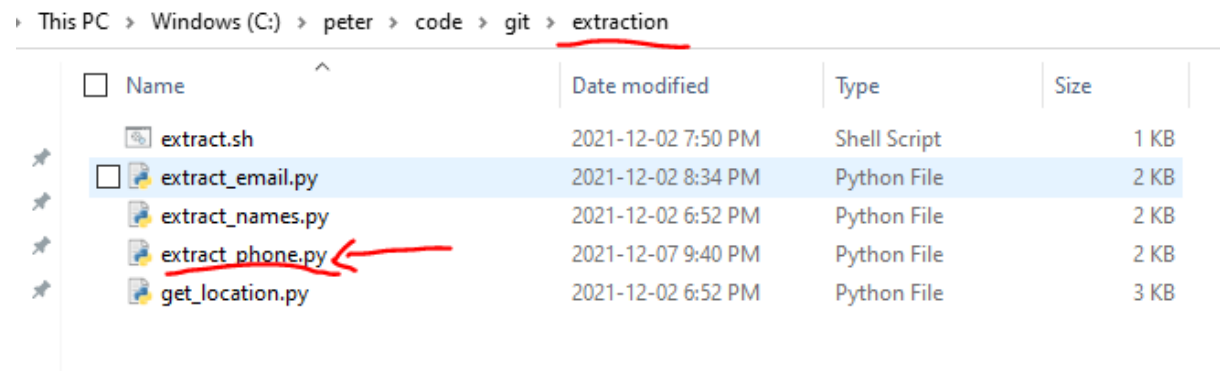
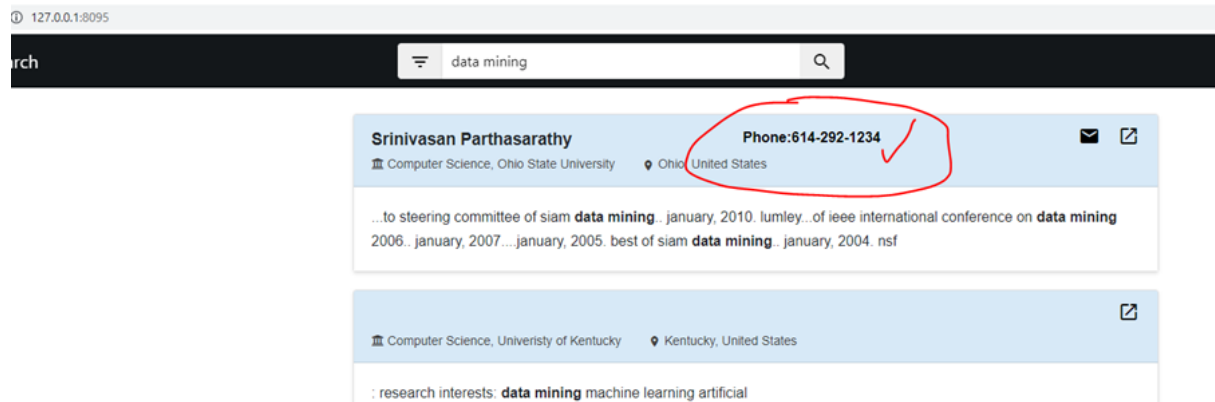
3.4.3 Significance

With building the classifier with a prediction accuracy of over 70%, we can help to realize automatic scraping faculty links from a random department link. We can also use this classifier in identifying whether a link is a wanted link or not only by changing the attributes.

3.5 Extract A Common Field

After discussing with the team, we decided to add the phone number into the web page as follows. To make this change, we researched the Flask web page solution, tested the end to end web solution, and made changes in a lot of places.

The new common field is displayed this way.



https://github.com/peterzhangon/Final410/blob/master/extraction/extract_phone.py

C > Windows (C:) > peter > code > git > data				
Name	Date modified	Type	Size	
compiled_bios	2021-12-07 9:41 PM	File folder		
expertsearch	2021-12-02 6:52 PM	File folder		
FacultyDataset	2021-12-02 6:52 PM	File folder		
filter_data	2021-12-07 9:40 PM	File folder		
.DS_Store	2021-12-02 6:52 PM	DS_STORE File	9 KB	
depts	2021-12-07 9:40 PM	File	149 KB	
emails	2021-12-07 9:40 PM	File	77 KB	
location	2021-12-07 9:40 PM	File	155 KB	
MP2_Part1 Signup - Sheet1.csv	2021-12-02 6:52 PM	Microsoft Excel C...	21 KB	
names.txt	2021-12-07 9:40 PM	Text Document	98 KB	
phones	2021-12-07 9:40 PM	File	32 KB	
unis	2021-12-07 9:40 PM	File	158 KB	
urls	2021-12-07 9:40 PM	File	966 KB	

<https://github.com/peterzhangon/Final410/blob/master/data/phones>

his PC > Windows (C:) > peter > code > git > static				
Name	Date modified	Type	Size	
bootstrap-4.0.0-dist	2021-12-02 6:52 PM	File folder		
.DS_Store	2021-12-02 6:52 PM	DS_STORE File	7 KB	
admin.css	2021-12-02 6:52 PM	Cascading Style Sheet D...	1 KB	
admin.js	2021-12-02 6:52 PM	JavaScript File	2 KB	
index.css	2021-12-02 6:52 PM	Cascading Style Sheet D...	3 KB	
index.js	2021-12-02 8:36 PM	JavaScript File	7 KB	
jquery-3.4.1.min.js	2021-12-02 6:52 PM	JavaScript File	87 KB	

<https://github.com/peterzhangon/Final410/blob/master/static/index.css>

3.6 Topic Modeling

PLSA and LDA have been shown excellence in performance especially in the field of NLP. However, they can be cumbersome to train and tweak. Here I leveraged BERT embedding, which stands for Bidirectional Encoder Representation from Transformers. It's designed to pre-train deep bidirectional representation from unlabeled text by jointly conditioning on both left and right context. As a result, the BERT model can be fine tuned for NLP tasks.

3.6.1 Embeddings

First step to convert data to numerical data; Subsequently I used sentence-transformers for embedding documents, you can choose whatever package you want. Then I chose to use distilbert for compromising speed and performance.

```

: from sentence_transformers import SentenceTransformer
  model = SentenceTransformer('distilbert-base-nli-mean-tokens')
  embeddings = model.encode(data, show_progress_bar=True)

```

3.6.2 Clustering

I wanted to make sure similar topics are clustered together. UMAP is used for high dimensional reduction while local structure in lower dimensionality.

```

umap_embeddings = umap.UMAP(n_neighbors=15,
                             n_components=5,
                             metric='cosine').fit_transform(embeddings)

```

Data are reduced to dimension of 5 and local neighborhood at 15. UMAP did not cluster anything but reducing dimensions. This package called HDBSCAN was used for clustering.

```

: import hdbscan
  cluster = hdbscan.HDBSCAN(min_cluster_size=15,
                             metric='euclidean',
                             cluster_selection_method='eom').fit(umap_embeddings)

```

3.6.3 Data Preparation & create TF-IDF score

```

: import matplotlib.pyplot as plt

# Prepare data
umap_data = umap.UMAP(n_neighbors=15, n_components=2, min_dist=0.0, metric='cosine').fit_transform(embeddings)
result = pd.DataFrame(umap_data, columns=['x', 'y'])
result['labels_'] = cluster.labels_

: docs_df = pd.DataFrame(data, columns=["Doc"])
  docs_df['Topic'] = cluster.labels_
  docs_df['Doc_ID'] = range(len(docs_df))
  docs_per_topic = docs_df.groupby(['Topic'], as_index = False).agg({'Doc': ' '.join})

```

$$c - TF - IDF_i = \frac{t_i}{w_i} \times \log \frac{m}{\sum_j^n t_j}$$

Where the frequency of each word t is extracted for each class i and divided by the total number of words w . This action can be seen as a form of regularization of frequent words in the class. Next, the total, unjoined, number of documents m is divided by the total frequency of word t across all classes n .

3.6.4 Topic representation

After we have an important value for each word in the cluster which can create the topic. If I take the top 10 most important words in each cluster, then we get a topic.

```
.]:
```

	Topic	Size
0	-1	9816
35	34	843
58	57	808
19	18	767
72	71	613
20	19	580
77	76	378
75	74	365
49	48	324
68	67	258

Topic size means how frequently certain topics appear. Topic name -1 refers to all documents that didn't have a topic assigned. This is a big thanks to HDBSCAN package which doesn't force every word towards a certain cluster. Our output data size from web scraping is relatively small compared to mainstream big data size. Hence the realistic Topic and Size looked like below.

```
[78]:
```

	Topic	Size
0	-1	12

```
[79]: top_n_words[-1][:10]
```

```
[79]: [('data', 0.22105663849357624),
      ('project', 0.17193515657377895),
      ('level', 0.14652345295965485),
      ('understanding', 0.14652345295965485),
      ('cell', 0.14652345295965485),
      ('systems', 0.14652345295965485),
      ('modelling', 0.14652345295965485),
      ('pathways', 0.12039518302700086),
      ('complex', 0.12039518302700086),
      ('signalling', 0.12039518302700086)]
```

Here we have top 10 significant words for a given topic. Yes, topic -1 meaning these words did not have a “topic assigned”. But because they are in the same group and they do have significant scores. We can still rely on the data. Top_n_words[-1][:10] means that for topic -1, the top 10 significant words within topic -1 are as such. The topic “data” has the highest score. Hence the topic is about Data.

3.7 Integration

To support future expansion, we created new scripts to preprocess the data files.

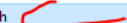
```

85 history
peter@LAPTOP-MSKOE3G5:/mnt/c/peter/code/git_local$ pwd
/mnt/c/peter/code/git_local
peter@LAPTOP-MSKOE3G5:/mnt/c/peter/code/git_local$ ./preprocess.sh 6524
Max txt file id: 6524
***** Begin to preprocess data.*****
/mnt/c/peter/code/git_local
+++++ Begin to extract information.+++++
Max txt file id: 6524
/mnt/c/peter/code/git_local


Begin to process emails, the source is at: /mnt/c/peter/code/git_local/data/compiled_bios
The destination is at : /mnt/c/peter/code/git_local/data/emails
File: /mnt/c/peter/code/git_local/data/compiled_bios/0.txt

```

In the preprocess file, we extract common fields like email, and phone numbers, and the existing functions are called also to update all the related data files like names, locs and so on.

gunicorn_config.py	2021-12-02 6:52 PM	Python File	0 KB
how_to_test.pdf	2021-12-02 8:54 PM	Adobe Acrobat Docum...	273 KB
 preprocess.sh	2021-12-02 7:50 PM	Shell Script	1 KB
README.md	2021-12-02 6:52 PM	MD File	1 KB
requirements.txt	2021-12-02 6:52 PM	Text Document	1 KB
server.py	2021-12-07 8:27 PM	Python File	8 KB
server.pyc	2021-12-02 8:13 PM	Compiled Python File	8 KB
write_file_names.py	2021-12-02 8:35 PM	Python File	5 KB
wsgi.py	2021-12-02 6:52 PM	Python File	1 KB

Windows (C:) > peter > code > git > extraction

Name	Date modified	Type	Size
 extract.sh	2021-12-02 7:50 PM	Shell Script	1 KB
extract_email.py	2021-12-02 8:34 PM	Python File	2 KB
extract_names.py	2021-12-02 6:52 PM	Python File	2 KB
extract_phone.py	2021-12-02 9:40 PM	Python File	2 KB
get_location.py	2021-12-02 6:52 PM	Python File	3 KB

To help the team to test the basic functions, we also created a doc named “how to test”, via which the team member can **know all the processes**.

4. Summary

With all the work described above, our team has done a remarkable job on the course project. We finished all the tasks we proposed, and contributed to the original project from both automatically scraping and extra information extraction.

This is for the presentation

Before the presentation,

Clone the latest code;
Go to the base directory, run:
./preprocess.sh 6524

Presentaion begins:

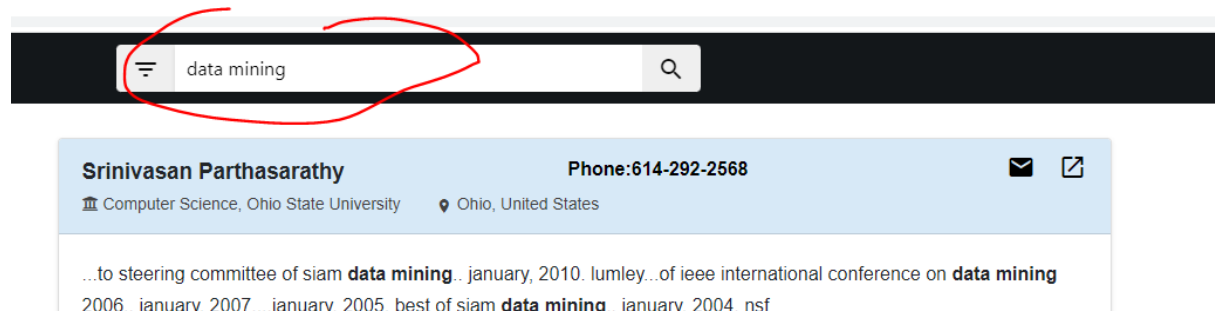
Now, I would like to briefly show you a demo.
First, let us start the web service

```
peter@LAPTOP-MSK0E3G5:/mnt/c/peter/code/git$ gunicorn server:app -b 127.0.0.1:8095
[2021-12-07 23:03:09 +0000] [64] [INFO] Starting gunicorn 19.7.1
[2021-12-07 23:03:09 +0000] [64] [INFO] Listening at: http://127.0.0.1:8095 (64)
[2021-12-07 23:03:09 +0000] [64] [INFO] Using worker: sync
[2021-12-07 23:03:09 +0000] [68] [INFO] Booting worker with pid: 68
```

Secondly, let us go to the home page.



let us input the query condition "data mining"



We can find the phone numbers are displayed, if we can extract them.

Srinivasan Parthasarathy
Phone:614-292-2568

Computer Science, Ohio State University
Ohio, United States

...to steering committee of siam **data mining**.. january, 2010. lumley...of ieee international conference on **data mining** 2006.. january, 2007....january, 2005. best of siam **data mining**.. january, 2004. nsf

Computer Science, Univeristy of Kentucky
Kentucky, United States

: research interests: **data mining** machine learning artificial

Computer Science, Univeristy of Kentucky
Kentucky, United States

...interests: computational medical imaging and **data** analysis bioinformatics **data mining**

Computer Science, Univeristy of Kentucky
Kentucky, United States

...computer science research interests: bioinformatics **data mining**.

Huan Sun
Phone:614-688-1078

Computer Science, Ohio State University
Ohio, United States

...ieee transactions on knowledge and **data** engineering (tkde 2015). tan,...ieee transactions on knowledge and **data** engineering (tkde 2014). presentations...."**mining** big **data** to build

Let us click this link to go to this Faculty's home page.

data mining

Srinivasan Parthasarathy
Phone:614-292-2568

Computer Science, Ohio State University
Ohio, United States

...to steering committee of siam **data mining**.. january, 2010. lumley...of ieee international conference on **data mining** 2006.. january, 2007....january, 2005. best of siam **data mining**.. january, 2004. nsf

We can see the phone number is as same as we displayed.

Srinivasan Parthasarathy



Professor, Computer Science and Engineering

Professor, Biomedical Informatics

Dreese Laboratories

2015 Neil Ave
Columbus, OH 43210-1210

(614) 292-2568

[parthasarathy.2](#)