Improving ExpertSearch Progress Report

[**Plan** 2](#_Toc57138365)

[**Automate scraping process** 2](#_Toc57138366)

[**Perform topic mining** 2](#_Toc57138367)

[**Additional improvement** 2](#_Toc57138368)

[**Improve UI** 2](#_Toc57138369)

[**Progress** 2](#_Toc57138370)

[**Automated scraping process** 2](#_Toc57138371)

[**Deliverables:** 2](#_Toc57138372)

[**Outputs:** 3](#_Toc57138373)

[**Automated Scraper** 3](#_Toc57138374)

[**Directory URL classification** 4](#_Toc57138375)

[**1.** **Dataset preparation** 4](#_Toc57138376)

[**2.** **Scraper** 4](#_Toc57138377)

[**3.** **Text classification** 5](#_Toc57138378)

[**Faculty URL classification** 5](#_Toc57138379)

[**1.** **Dataset preparation** 5](#_Toc57138380)

[**2.** **Scraper** 5](#_Toc57138381)

[**3.** **Text classification** 6](#_Toc57138382)

[**Topic Mining** 6](#_Toc57138383)

[**Deliverables:** 6](#_Toc57138384)

[**Improved Email Extraction** 6](#_Toc57138385)

[**Deliverables:** 6](#_Toc57138386)

[**UI Improvements** 6](#_Toc57138387)

[**Deliverables:** 6](#_Toc57138388)

# **Plan**

## **Automate scraping process**

* To identify faculty directory pages
* To identify faculty home pages

## **Perform topic mining**

* To identify top-k topics associated with each faculty

## **Additional improvement**

* To improve email extraction for each faculty

## **Improve UI**

* To display top-5 topics associated with each retrieved faculty
* To allow search based on any of the topics from the displayed topic cloud
* To prepopulate email content when clicked on a faculty’s email address

# **Progress**

|  |  |  |
| --- | --- | --- |
| **Item** | **Owner** | **Status** |
| Automated Scraping | Mriganka Sarma | Completed items:   * Automated scraper * Data Handler * Scraper * Text Classifier |
| Topic Mining | Zacharia Rupp |  |
| Improved Email Extraction |  |  |
| Improved UI | Sai Ranganathan |  |

More detailed description is provided in the below sections.

## **Automated scraping process**

### **Deliverables:**

* Automated Scraper (auto\_scraper.py)
* Data Handler (data\_handler.py)
* Scraper (scraper.py)
* Text Classifier (text\_classifier.py)

### **Outputs:**

* Corpus of classified Faculty Directory URLs
* Corpus of classified Faculty Bio URLs
* Documents of bios for each faculty generated by scraping the classified Faculty Bio URLs

### **Automated Scraper**

Automated scraper module (auto\_scraper.py) automates the process in the following way:

* Uses the data handler to prepare a train and test set of Faculty Directory URLs
* Uses the scraper to scrape these URLs to prepare the train and test corpus
* Uses the text classifier to build and train a Doc2Vec model on the documents in the train corpus of directory contents
* Uses the text classifier to predict the category of the test URLs as “Directory” or “Non-Directory”
* Saves the classified directory URLs to a file
* Uses the data handler to prepare a train and test set of Faculty Bio URLs
* Uses the scraper to scrape these URLs to prepare the train and test corpus
* Uses the text classifier to build and train a Doc2Vec model on the documents in the train corpus of faculty bios
* Uses the text classifier to predict the category of the test URLs as “Faculty” or “Non-Faculty”
* Saves the classified bio URLs to a file
* Uses the scraper to scrape the faculty bios from the classified bio URLs
* Generates one document per faculty bio and saves under ExpertSearch/data/compiled\_bios

The automated scraper can be invoked as follows:

**$ cd ExpertSearch/AutoScraper**

**$ python ./auto\_scraper.py -d -t**

-d option specifies to generate/regenerate the train and test dataset.

The dataset will be generated even if -d is not provided if the dataset doesn’t exist yet.

When -d is not provided, the existing dataset will be used.

-t option specifies to retrain the Doc2Vec model on the train dataset.

The model will be trained even if -t is not provided if the model wasn’t trained and saved yet.

When -t is not provided, the saved model will be loaded.

The following sections describe the text classification tasks for Faculty Directory URLs and Faculty Bio URLs.

### **Directory URL classification**

#### **Dataset preparation**

First we need to prepare the dataset for training and testing the model. The following approach was used to prepare the dataset.

* + Downloaded the known faculty directory pages from the sign-up sheet for MP 2.1. These will serve as the “positive” examples.
  + Collected top URLs from Alexa. These will serve as the “negative” examples.
    - Collected the global top-50 pages of Alexa.
    - Collected the top-50 pages for different countries. Manually verified that the pages are in English.
  + About 900 URLs were obtained from the sign-up sheet data, which was partitioned into 500 for training and 400 for test data.
  + URLs for total 14 countries + top-50 global URLs from Alexa were collected. This gave 750 “negative” URLs.
  + Wrote a python module (data\_handler.py) for data handling that does the following:
    - Converts the MP 2.1 sign-up data from csv to a file containing only the directory URLs. Performs any cleanup as necessary and labels them as “directory”.
    - Combines the top-50 Alexa URLs for 10 countries and labels them as “alexa\_dir”. Uses these 500 pages for training.
    - Combines the top-50 Alexa URLs for 5 countries and labels them as “test\_dir”. Uses these 250 pages for testing.
    - Mix the 500 Faculty Directory training URLs with the 500 Alexa training URLs. Remove duplicates if any. This gives 734 URLs as the final training URLs.
    - Mix the 400 Faculty Directory test URLs with the 250 Alexa training URLs. Remove duplicates if any. This gives 548 URLs as the final test URLs.

#### **Scraper**

Wrote a python module (scraper.py) for scraping the URLs collected from the above step.

The scraper does the following:

* + Gets the contents of each URL as text.
  + Performs clean-up of non-ascii characters from the content.
  + Performs other clean-ups such as substituting newlines, tabs, multiple whitespaces into single whitespace.
  + Substitutes contents such as “403 Forbidden”, “404 Not found”, etc. with “Error: Content Not Found”.
  + Writes contents of each webpage as a single line of space separated words in a file meant to be the final corpus.
    - This is done to prepare both the training corpus (“train\_dataset.cor”) and the test corpus (“test\_dataset.cor”).

#### **Text classification**

Wrote a python module (text\_classifier.py) for performing the text classification task of identifying valid Faculty Directory pages from the test corpus.

The classification module does the following:

* + Uses gensim to build a Doc2Vec model for feature vector representation of each document.
  + Uses the train\_dataset.cor to build the vocabulary and train the model.
  + Saves the model so that it can be reloaded while running next time on the same dataset.
  + Uses LogisticRegression as the classifier from scikit-learn module.
  + Uses LogisticRegression to predict the categories of the test URLs given the test dataset.

### **Faculty URL classification**

#### **Dataset preparation**

The following approach was used to prepare the dataset:

* + Use the top 1000 URLs from the currently existing Faculty Bio URLs in the ExpertSearch project as the train URLs.
  + Use 250 URLs from the Alexa test URLs set as the “negative” train URLs.
  + Use the data handling module to do the following:
    - Tag the faculty bio URLs as “faculty” and save to a file.
    - Tag the Alexa URLs as “alexa\_faculty” and save to the same file.
    - This will be the final file with all the train URLs.

#### **Scraper**

Since the ExpertSearch project already contains the faculty bios as documents, the top 1000 faculty bios are copied to the train corpus file (“train\_bio\_dataset.cor”).

Then the scraper does the following:

* + Scrapes the remaining train URLs from the train URLs file and appends to the train corpus (“train\_bio\_dataset.cor”).
  + Uses the classified Faculty Directory URLs from the classified Directory URLs file above and gets all embedded potential faculty bio URLs as the test URLs.
  + Scrapes the test URLs from above and adds to a test corpus (“test\_bio\_dataset.cor”).

#### **Text classification**

The classification module does the following:

* + Uses gensim to build a Doc2Vec model for feature vector representation of each document.
  + Uses the train\_bio\_dataset.cor to build the vocabulary and train the model.
  + Saves the model so that it can be reloaded while running next time on the same dataset.
  + Uses LogisticRegression as the classifier from scikit-learn module.
  + Uses LogisticRegression to predict the categories of the test URLs given the test dataset.

## **Topic Mining**

### **Deliverables:**

## **Improved Email Extraction**

### **Deliverables:**

## **UI Improvements**

### **Deliverables:**