Project Documentation:

Web Scraping

1. Scrapy Framework –
    a. Utilized scrapy framework with Python to crawl web pages and identify connected links.
2. Beautiful Soup –
    a. Utilized BeautifulSoup toolkit with Python to extract text from web pages.
    b. Removed special characters using regular expressions and extracted text data from web pages.

Classification:

1. Once the web pages are scraped using the 'Crawl-n-Extract' module (see run help below). The web pages text is saved in a raw text file with each web page text on one line (similar to MP2 exercise).
2. The classification task is to classify a given webpage as a Faculty Page (positive class: 1) or Non Faculty Page (negative class: 0).
3. We trained multiple classifiers as part of this exercise and results (F1-score and accuracy) can be found in our presentation.
4. We have built most of the model based on the text classification techniques that we have learnt as part of the course. We have mainly used Tf-Idf vectorizer and Bert based sen2Vec to vectorize the text of the web pages.

- **Logistic Regression**

    1. Used Scikit learn Logistic regression module as well as scikit learn tf-idf vectorizer for vectorizing the web pages. We have
    2. Used nltk library to remove stopwords and for stemming.
    3. F1-Score (hold out test set): **0.9702970297029703**

- **XGBoost**

    1. Used XGBoost library for training the model
    2. Scikit learn tf-idf vectorizer for vectorizing the web pages
    3. F1-Score (hold out test set): **0.9829683698296837**

- **Deep learning**
    1. Tensorflow –
       Used tensorflow to build the deep-learning model.
        a. Four layered neural network model(excluding input layer)

b. Uses Adam optimizer with learning **0.0001**
c. Loss is evaluated using **Binary Cross Entropy**
d. F1 Score on test data set: **0.9963**

2. NLTK –
   Utilized NLTK to pre-process the data. Pre-processing involved following list of steps
   a. Remove stop words
   b. Stemming

- **(Experimental) BERT encoding based Logistic Regression**
  1. Generated the embeddings for the web pages text using a pre-trained BERT model ('`distilbert-base-nli-mean-tokens`')
  2. Used the transformer library to generate the embeddings
  3. Built an experimental Logistic regression model
     a. F1 Score for this experimental model is: **0.9874055415617129**
  4. As, we can see there is an improvement over the Tf-IDF based Logistic Regression model, hence this experiment was successful. But we were getting better results with our custom neural network model, we didn't pursue this further. In future, we would like to expand on this idea of transferring knowledge (from BERT encoding) to inform and improve simple classifiers. This saves a lot of time as the pre-trained models are generally trained on large corpus of wikipedia/web pages and will provide a decent knowledge base to transfer to a simple classifier model which then can be trained much easily in a short span of time.

Topic Modeling on Faculty bios

➔ Used gensim library to generate the topic model based on the compiled bios
➔ Running information can be found in '**Run Help**' section

NER Model

➔ Used spacy library to extract named entities, in particular faculty names and different organizations that are mentioned on their page. Both these entities will help in improving the search index.

Run help:

Python setup (*python >= 3.5, only web scraping needs python 3.5, other modules work with newer versions of python also***):**

Please run : `pip install -r source_code/requirements.txt`
`You can also use the venv by using following commands:`

- `cd source_code`
- `python -m venv <venv-name>`
- `source <venv-name>/bin/activate`
- `python3 -m pip install -r requirements.txt`

Web Scrapping:

1. **Scrapy (Use Python 3.5 Version):**
   - **a.** Script created using Scrapy framework will extract links for the web-page
   - b. Create a scrapy project using
     - i. "Scrapy startproject link_extractor
   - c. Configure items.py to create class "LinkExtactorItem" class and define below fields for the item
     - i. url_from
     - ii. urk_to
   - d. Create a new spider using below command
     - i. Scrapy genspider uiuc
   - e. Crawl links of a web-page using below command
     - i. scrapy crawl uiuc -o links.csv -t csv

     It will extract all the links associated with web page to csv file.
2. **Split (Use Python 3.5 Version)**
   - a. Split the input file into multiple files using script "split.py"
3. **Extract web contents (Use Python 3.5 Version)**
   - a. Extract web contents using script "python extract.py uiuc1.csv uiuc1.txt". This script uses Beautiful soup as the toolkit
4. **Merge (Use Python 3.5 Version)**
   - a. Merge all output files using script "merge.py"

Web Page Classification

Logistic Regression Model (python >= 3.7):

➔ **Training:**

- Navigate to `source_code/logistic_regression`
- Run `python train.py`

- The model file will be saved at `source_code/logistic_regression/logit.model`

➔ **Inference:**

- There are two ways to run inference once:
  - By providing a file with all the webpage (one str per web page on one line):
    - From root directory (TextInformationSystem-CourseProject) run following command: `python -m source_code.logistic_regression.inference <webpage data file path>`
      - `For file format look at: TextInformationSystem-CourseProject/source_code/Crawl-n-Extract/Merge/UIUC.txt`
  - By FacultyClassifier module (which is an entry point for all of our classification models):
    - Create an object of FacultyClassifier class with classifier_type = 'logit' and then run predict method on list of web pages text
    - A sample run is defined in the "__main__" block of that module.

XGBoost Model (python >= 3.7):

➔ **Training:**

- Navigate to `source_code/XGboost`
- Run `python train.py`
- The model file will be saved at `source_code/XGboost/xgb.model`

➔ **Inference:**

- There are two ways to run inference once:
  - By providing a file with all the webpage (one str per web page on one line):
    - From root directory (TextInformationSystem-CourseProject) run following command: `python -m source_code.XGboost.inference <webpage data file path>`
      - `For file format look at: TextInformationSystem-CourseProject/source_code/Crawl-n-Extract/Merge/UIUC.txt`
  - By FacultyClassifier module (which is an entry point for all of our classification models):
    - Create an object of FacultyClassifier class with classifier_type = 'xgb' and then run predict method on list of web pages text
    - A sample run is defined in the "__main__" block of that module.

Neural Network Model (python >= 3.7):

➔ **Predict Faculty Pages (Inference):**

Trained neural network model is of size 187MB on disk. Git has an upper cap of 100MB. For this reason, the trained model has been uploaded to box. Please download the trained neural network model from the box link below

Step 1: Download the folder 'neural_network_model_v2' from the link below.
https://uofi.box.com/v/es-neural-network-model-v2

Step 2: Ensure that directories 'model', 'vectorizer' are placed directly under 'fully_trained_model'.

Step 3: Ensure below directories are safely extracted.
'fully_trained_model/model'
'fully_trained_model/vectorizer'

Step 4: Point 'crawled_data_path' in the script 'classify/infer_crawled_data.py' point to your webpage dump. **Note:** The crawled data should have one doc per line. Separated by '#####'
(**Eg:** ~/source_code/Crawl-n-Extract/Merge/UIUC.txt)

Step 5: Run the script 'infer_crawled_data' and observe the faculty links printed in the console.

Note: *You can also use source_code/faculty_page_classifier/faculty_page_classifier/FacultyClassifier to programmatically use this model to run predictions. You need to initialize the FacultyClassifier with classifier_type='nn'. But this also requires copying over the model file from the location provided in Step 1 and saving it to the appropriate folder.*

➔ **Training the model:**

We suggest that you use the model we have already trained. Please follow the steps below if you intend to train a new model using the source code.

- Use the script "model_driver.py" under the package below to train the model "source_code.neural_network_ml_classifier.train"

- Initialize the path to training data and target location
    a. data_base_dir - base directory to training data. This data must have been preprocessed using PreProcessor.py
    b. The script then splits the data into 70% train and 30% validation dataset

c. This data used then used to train the four layered Neural network model

Topic Modeling on Faculty bios

- ➔ Used gensim library to generate the topic model
- ➔ Topic model can be run using the following notebook (this notebook contains self explanatory documentation to run the notebook):
  - ◆ `source_code/topic_modeller/TopicModelling.ipynb`

NER Model

- ➔ Used spacy library to extract named entities, in particular faculty names and different organizations that are mentioned on their page. Both these entities will help in improving the search index.
- ➔ NER Model can run using following notebook (this notebook contains self explanatory documentation to run the notebook):
  - ◆ `source_code/ner_model/spacy_ner.ipynb`
- ➔ Also, extracted faculty names as well as different organizations mentioned on the faculty pages. These two data points will help in indexing the faculty bios for searching.
  - ◆ Commands to run this step:
    - `cd source_code/ner_model`
    - `python extract_names_and_orgs.py <bios_dir> <destination_dir_for_names> <destination_dir_for_orgs>`
    - `E.g. bios_dir = source_code/data/compiled_bios, destination_dir_for_names = source_code/data/compiled_bios_names, destination_dir_for_orgs = source_code/data/compiled_bios_orgs`
- ➔ Names are extracted and saved in source_code/data/compiled_bios_names/
- ➔ Organizations are extracted and saved in source_code/data/compiled_bios_orgs
- ➔ We have used the same filenames as faculty bios for easier retrieval when this information is needed later