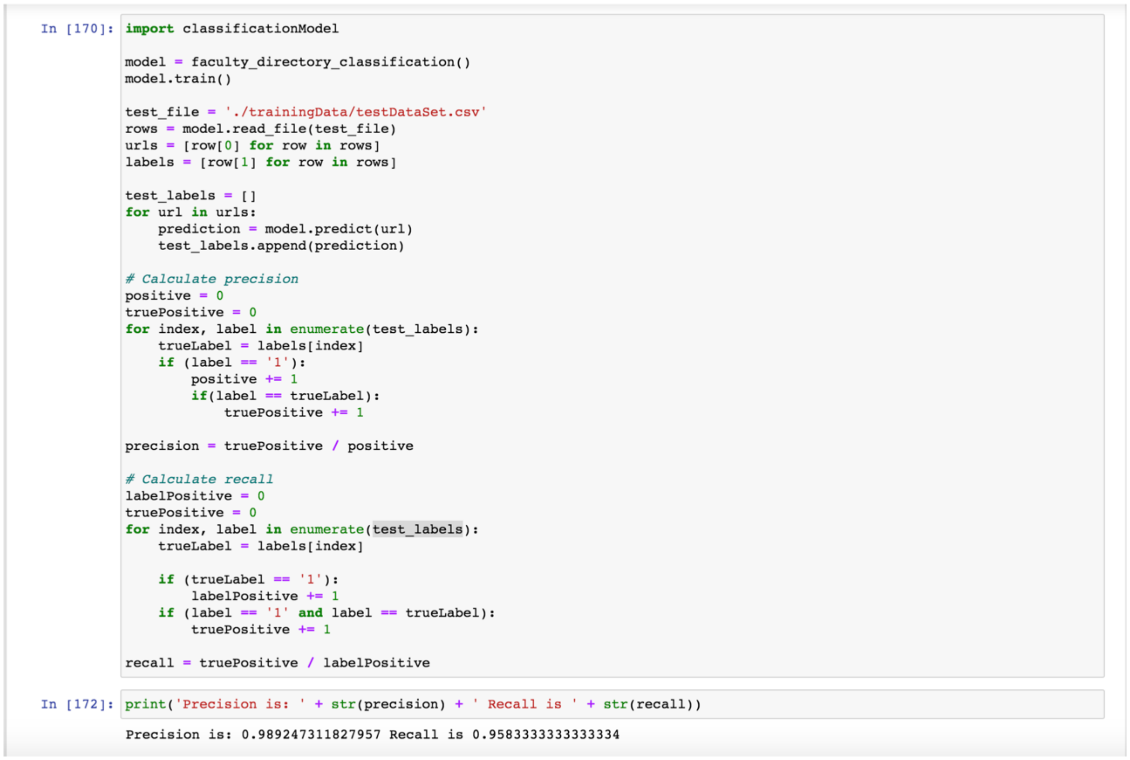
CS410 Final Report – Automatically crawling faculty webpages

This crawler can be used to automatically fetch faculty webpage URLs given a university website. It is designed to use text classification to identity relevant URLs.

Solutions

1. Identifying faculty directory pages and faculty webpage URLs

   Built a Support Vector Machine(SVM) model to classify a URL into a directory page vs. non-directory page. The training data is listed in the trainingData folder and the model is defined in the classificationModel.py file. The idea is to split the words in the URL and use these words as the feature matrixes to feed into the model. After the model is trained using the Python sklearn library, tested the model with another 107 testing data set. The testing process and the precision/recall scores is attached below.



```
In [170]: import classificationModel

          model = faculty_directory_classification()
          model.train()

          test_file = './trainingData/testDataSet.csv'
          rows = model.read_file(test_file)
          urls = [row[0] for row in rows]
          labels = [row[1] for row in rows]

          test_labels = []
          for url in urls:
              prediction = model.predict(url)
              test_labels.append(prediction)

          # Calculate precision
          positive = 0
          truePositive = 0
          for index, label in enumerate(test_labels):
              trueLabel = labels[index]
              if (label == '1'):
                  positive += 1
                  if(label == trueLabel):
                      truePositive += 1

          precision = truePositive / positive

          # Calculate recall
          labelPositive = 0
          truePositive = 0
          for index, label in enumerate(test_labels):
              trueLabel = labels[index]

              if (trueLabel == '1'):
                  labelPositive += 1
              if (label == '1' and label == trueLabel):
                  truePositive += 1

          recall = truePositive / labelPositive

In [172]: print('Precision is: ' + str(precision) + ' Recall is ' + str(recall))

          Precision is: 0.989247311827957 Recall is 0.9583333333333334
```

2. Built the Crawler

   A demo of this crawler can be found at https://github.com/yunhezhang/CourseProject/blob/main/demo.mp4. The source code is in crawler.py.

   The web crawler is based on BeautifulSoup and Selenium web driver and leverage the trained SVM model to identify faculty webpages. Given a university website, it first grabs all the faculty directory URLs predicted by the trained SVM model, then it loops through these directory URLs to grab all the faculty webpage URLs predicted by the same model until we reach a pre-set maximum number.

Set up Instructions

- Clone the repo
- In the repo directory, open Jupyter Notebook (I'm using Python3) and create a new notebook or use the existing demo notebook at https://github.com/yunhezhang/CourseProject/blob/main/src/demo_notebook.ipynb
- Run the steps Similar to the demo notebook
    - from crawler import crawler # May need to install required python packages at this step
    - crawler = crawler() # Note that chromedriver need to be installed in this directory. This step is to initialize the crawler, which sets up the trained SVM model and the web driver
    - crawler.crawl_directory_url('https://www.cs.rutgers.edu') # Given a university website, returns all the possible directory URLs
    - crawler.crawl_faculty_url('https://www.cs.rutgers.edu', 'https://www.cs.rutgers.edu/people/professors') # Given a university website and a directory URL, returns all the possible faculty webpage URLs
    - crawler.crawl('https://www.cs.rutgers.edu') # Combine the above two crawling steps into one. Given a university, returns the faculty webpage URLs up to the pre-set maximum number