Text Information Systems : CS410

Project Proposal

“2.2 ExpertSearch System”

--

Navyaa Sanan (navyaas2)
Srivardhan Sajja (sajja3)

Team

- **Navyaa Sanan** (navyaas2)
- **Srivardhan Sajja** (sajja3) : Team Captain

System

2.2 **ExpertSearch System**: A*utomatically crawling faculty webpages*

We plan on augmenting the ExpertSearch system by adding functionality which would make the system automatically crawl through faculty webpages given the primary university link/url (illinois.edu, berkeley.edu, etc.), instead of having to explicitly identify them.

Our project will have two main components. First, we plan to implement our own classifier, which given any URL will use text classification techniques mentioned in this course to judge whether the given URL is that of a faculty directory page. Second, given a primary university link we will find all directory pages associated with that primary URL. We will be using the classifier built in part 1 for implementing part 2.

Datasets, Algorithms and Techniques

Datasets: To train the extension, we shall be using the starting half of urls list fro mthe Google Sheets spreadsheet of MP2.1 as positive examples, and automatically generated non-faculty directory pages of university websites as negative examples. Similarly, to test our extension, we shall be using the 2nd half of the urls in the spreadsheet for positive examples, and randomly generated real urls as negative examples.

Algorithms/Techniques: To get the list of all urls from a university website, we shall use simple web crawling techniques using the built-in python packages but might dip into external packages if required. To make the classifier, we shall be using basic classification techniques and algorithms taught in this course. If we feel the need to use algorithms and techniques that are not mentioned in lectures, we will be sure to use techniques taught to us in the MPs. Overall, we will ensure that any algorithms used by us are directly or indirectly related to this course.

Primary Function

We are adding a function to automatically compile a list of all faculty directory pages from a given university home page url. This can be tested by comparing the generated list of urls with a list of manually compiled faculty directory page urls from different universities. The closer the pre-compiled list is to the generated one, the higher the accuracy of the web crawler and classifier is.

Communication with the system

Our extension is independent of the system. We consider this project to be an independent feature addition, which is inspired by the goal of this project but does not directly rely on any preexisting code. We will draw inspiration from MP2 and use the techniques taught to us in this course, but we will not have any direct reliance on any preexisting code whatsoever.

Programming Language

We plan to use Python3 for all developmental activities, including building the classifier and the web crawler.

Work Justification

Since there are 2 team members in this group, we planned this project to be 40 hours of work. Here is a breakdown of all tasks with the estimated time we expect each task to take:

1. Developing a classifier which, given a URL tells us if the URL is that of a faculty directory web page (23 hours)
    a. Subtask 1: Research and enlist at least 200 types of faculty directory URLs (2.5 hours)
    b. Subtask 2: Determine common/ identifying features of these faculty directory URLs (3 hours)
    c. Subtask 3: Look at research papers/scholarly texts that deal with similar classification problems and choose a classifier accordingly. Potentially write some pseudo code/ starter code for our classifier (7 hours)
    d. Subtask 4: Develop a training and testing data set and manually flag all websites we use. (2.5 hours)
    e. Subtask 5: Write the code of our classifier based on the reference. (5 hours)
    f. Subtask 6 : Based on the results, modify and fine tune our classifier. (3 hours)

2. Given a primary url (eg. Illinois.edu) identify all possible faculty directory pages. (17 hours)
    a. Subtask 1: Make a training testing data set of at least 10 primary URLs and as many directory URLS associated with the primary URL (2 hours)
    b. Subtask 2: Identify all the possible directory page associated URLs with our training/testing data set. (3 hours)
    c. Subtask 3: Read research papers / scholarly literature on how to tell whether a given URL is valid or not (4 hours)
    d. Subtask 4: Devise a method to find all valid URLs associated with the given primary URL. (5 hours)
    e. Subtask 5: Find a way to determine which subset of valid URLs are faculty directory pages. (should be just using the classifier that we built in Task 1 but since generating all possible combinations and testing out whether they are valid or not can take time, 3 hours to run the whole thing)