CS 410 FA20 - Project Proposal

2.2 ExpertSearch System - *Extracting relevant information from faculty bios*

Keon Park, keonp2@illinois.edu (Individual; Captain)

Overview

For my project, I plan to build upon ExpertSearch's extraction features and develop a new system that, given a faculty webpage's URL queried by the user, extracts not only names and emails but also other relevant information such as research and instruction areas by implementing keyword extraction on the text data. These extracted keywords will function as "tags" to efficiently indicate to the user the faculty member's top topic areas.

Impact

My system will tackle the problem of converting unstructured text into more structured data by providing a new means of effectively and quickly summarizing faculty text data into topics/keywords for users to more easily understand, instead of having users manually visit and read faculty webpages for more information outside of just names/emails like ExpertSearch. To demonstrate my system's usefulness, I will compare its results (containing structured information and topics) with ExpertSearch's results (basic word matches) for a given set of UIUC faculty.

Architecture

My new system will be implemented as a web application, with a React.js frontend and Python-Flask backend. Through the frontend, the user will be able to enter a URL of a faculty webpage, which the backend will then scrape and extract information from. Although my system will be separate from the existing ExpertSearch system, I plan to use its faculty name/email extraction code as a starting point/reference for my own implementation.

Data / Techniques

I plan to mainly use UIUC faculty and their webpages during development to test my system. For scraping text from faculty webpages, I will use my work from MP2.1 as a reference. To help implement my extraction techniques, I plan to use existing libraries such as NLTK and TextBlob.

Key Tasks + Timeline

1. Build web scraper with ability to detect and extract basic information from faculty profile sections (name, title, research, bio, contact, etc), using NER and HTML tags - 5 hours
2. Implement keyword extraction to tag faculty to relevant topics (research areas, areas of expertise, etc) - 4 hours
3. Test and refine web scraper and extraction functionality - 2 hours
4. Incorporate code from steps 1-3 into Flask backend API - 2 hours
5. Implement frontend for faculty webpage URL queries and displaying results - 6 hours
6. Integrate backend with frontend and finalize system - 3 hours