Comp 479/6791 **Project 2** **Fall 2025**

Objectives: Experiment with web crawling, scrape and index a set of documents. Cluster using k-means.

Due Date: 24. November 2025 (this is a change form the Course Outline)

Goal Extract Master's and PhD theses as well as other .pdf documents from Concordia Open Access portal Spectrum. This requires to read and understand the html structure of Spectrum. Required steps are to use a spider to find the .pdf documents one at a time, to pipe the document through Beautiful Soup to extract the text, them make the text into a token stream and add that to the inverted index for Spectrum. Then the text can be discarded and the next text can be processed.

Lab support To get to the documents requires some skill that I did not teach you. The lab instructors will support that.

Description and marking:

- starting from page `https://spectrum.library.concordia.ca/`, crawl for links within the `library.concordia.ca` domain (you may find crawling tools at `https://github.com/BruceDone/awesome-crawler`). A crawler that extracts text is acceptable. If your chosen crawler extracts URLs only, consider BeautifulSoup (`https://www.crummy.com/software/BeautifulSoup/`) to extract the text (but note `https://stackoverflow.com/questions/68777334/scraping-pdfs-from-multiple-pages-using-bs4`. Also see `https://www.geeksforgeeks.org/downloading-pdfs-with-python-using-requests-and-beautifulsoup/`. Guidance will be given in the labs). Describe and attribute any tools used in your Report (5pt, Attrib 5)
- make sure you obey the standard for robot exclusion (`https://www.robotstxt.org` covers robots.txt and meta tags) (-1pt, if not implemented)
- your crawler must accept as part of its input an upper bound on the total number of files to be downloaded. In developing, testing, and debugging, this number should be kept as small as possible. Develop your own closed test set for testing and debugging. (-1pt, if parameter not implemented)
- compile an inverted index. It is strongly suggested that your pipeline goes from landing on a page to piping it through Beautiful Soup or similar and adding its tokens immediately to the index, so that the document does not have to be stored on your system (or only temporarily). Note that it is important to save the resulting index to a file (like an "index.pkl" or "index.json") for further reuse in the following steps (2pts, Attrib 5)
- design queries to collect Spectrum documents on the queries *sustainability* and on *waste*
- for both collections, report how many documents you returned and how many documents were returned for both returns. Create a new collection *My-collection* that contains all documents from both queries (without duplicates)
- use scikit learn to cluster the resulting document collection as represented by your index (see `https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html#sphx-glr-auto-examples-text-plot-document-clustering-py`) (5pt, Attrib 5)
- run three different clustering runs, namely for $k = 2$, $k = 10$, $k = 20$. Try to assess the clusters - do they make sense to you? See what 'name' you would give the clusters? Compile the top 50 vocabulary terms for each cluster ranked by tf/idf. (5pt, Attrib 6) [1 bonus point: Consider two document frequencies: once within the entire index collection, once within *My-collection*. What is the difference on the clusters you get?]
- report on the different behaviour of the three clusterings and your experience with crawling and scraping of web pages. Limit your Report to 5 pages (5pts, Attrib 6)
- compile a demo file, where you present walkthroughs for the markers highlighting good and bad example clusters (3pt, Attrib 6)

Deliverables:

- 'code' for the markers to rerun. Give the sequence of calls to the different packages with all parameters in a separate file called README. Make sure that urls to the package version you use are part of your in-line comments
- separate results for each clustering (find a way to represent it during lab time). Print out 20 index terms for each cluster that you find most informative and indicate their tf/idf ranks
- one .pdf file called *Report*, including your design choices, findings, and anything interesting that the marker should pay attention to
- one demo file called *Demo* that showcases your efforts and insights