UIUC MCS-DS Coursework Selector

Project Goal

The aim of this project is to streamline course selection, while meeting graduation requirements, for MCS-DS online students through a single interface. Previously, students would need to flip between degree requirements and course description pages, parsing through the content to determine if a course was interesting and met their needs for degree completion. Now, users will be able to select any topic interests and then a list of courses needed to complete their degree will be presented to them.

Software Requirements

This selector has 5 main components. First, a scraper parses MCS-DS degree requirements from the web and store them and their acceptable coursework. Second, a crawler grabs course descriptions for relevant courses in the course catalog, where 1 course description = 1 document. Third, the documents are cleaned and processed into a bag-of-words representation to allow for the creation of a tokenized inverted index. Fourth, probabilistic topic mining is used on the documents to create a list of topic interests for the user to select from. Fifth, a course selector function compares the user selected interests against the inverted index and cross-references them with the degree requirements.

User Guide

Follow the steps in this guide to setup and use the UIUC MCS-DS Course Selector.

Required Packages

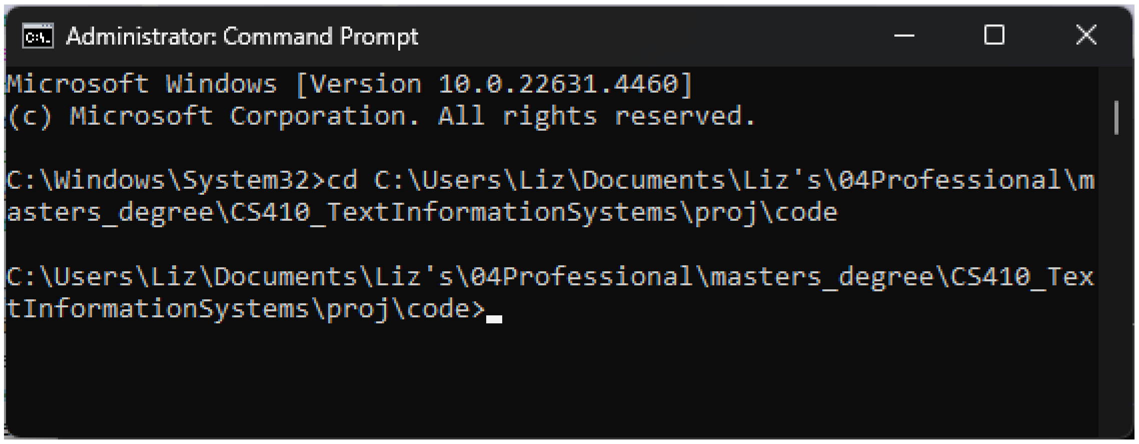
- Python (3.10.11 or higher)
- Pip (24.3.1 or higher)

Setup

1. Download program files from GitHub into a single folder.
2. Ensure python is setup properly on your PC by following the instructions at https://wiki.python.org/moin/BeginnersGuide/Download.
3. Ensure pip is setup properly on your PC by following the instructions at https://pip.pypa.io/en/stable/getting-started/.
4. Any other dependencies should be loaded during program execution

Program Execution

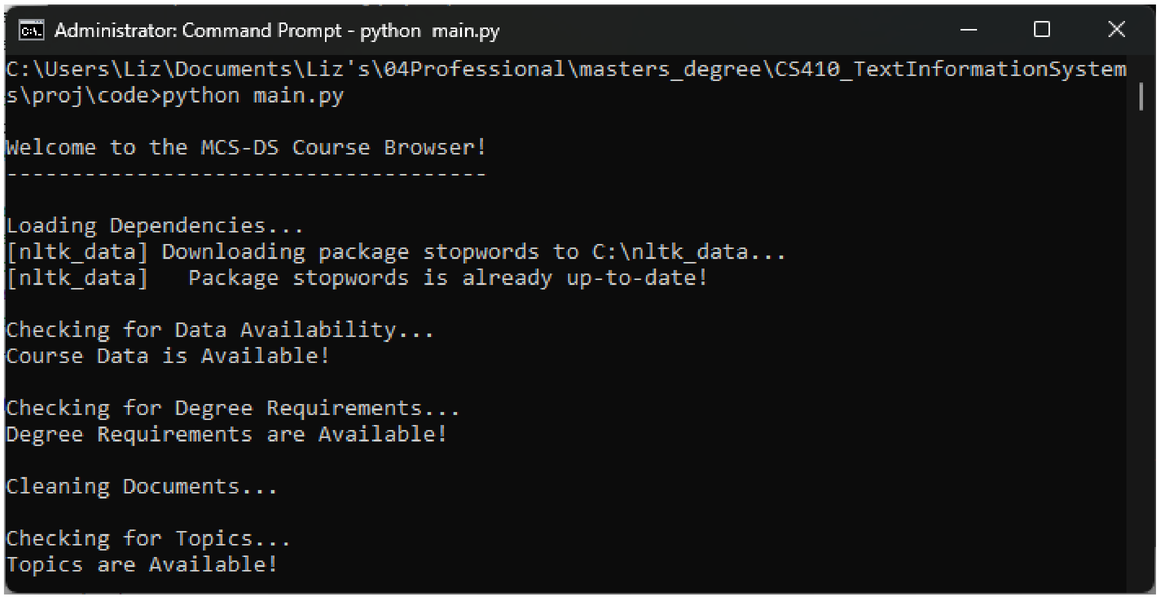
1. Run a command prompt as Administrator
2. Navigate to the program folder using *"cd"* command



Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\Liz\Documents\Liz's\04Professional\m
asters_degree\CS410_TextInformationSystems\proj\code

C:\Users\Liz\Documents\Liz's\04Professional\masters_degree\CS410_Tex
tInformationSystems\proj\code>_
```

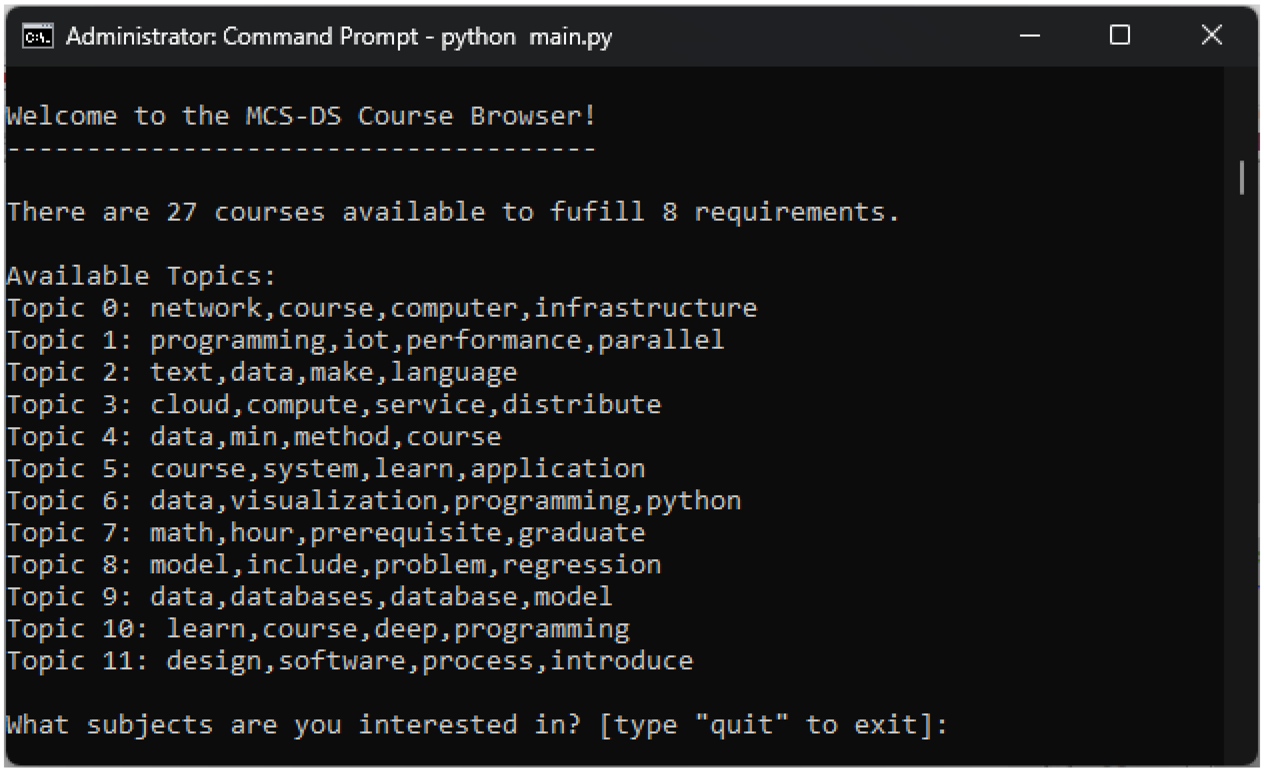
3. Run main.py using python. For example, *"python main.py"*. Available data will be loaded.
   a. If Dependencies fail to load, try running installation command separately.
      i. Ensure program folder is on path for pip
      ii. Run installation command using python *"pip install -r requirements.txt"*
   b. If data, requirements, or topics are not found, answer prompts "yes" to retrieve them. WARNING: These steps will take extra time.



Administrator: Command Prompt - python main.py

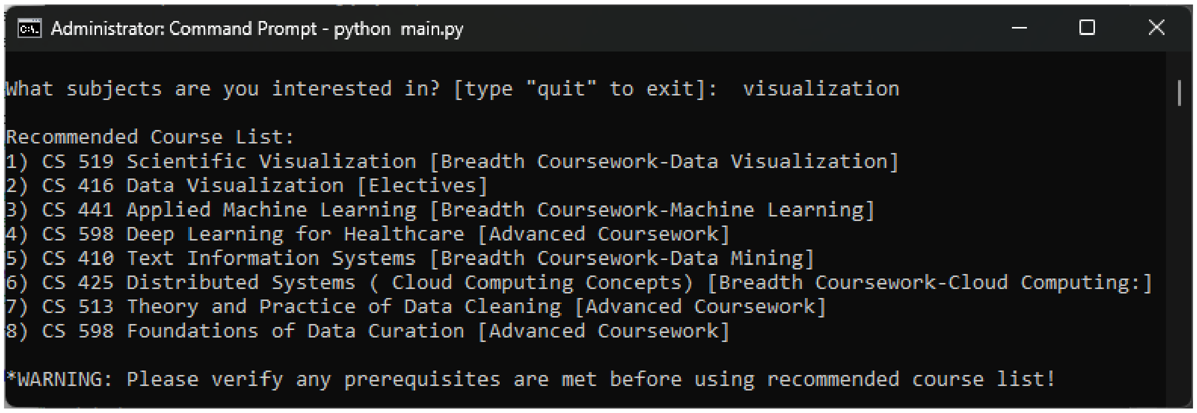
```
C:\Users\Liz\Documents\Liz's\04Professional\masters_degree\CS410_TextInformationSystem
s\proj\code>python main.py

Welcome to the MCS-DS Course Browser!
-------------------------------------

Loading Dependencies...
[nltk_data] Downloading package stopwords to C:\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Checking for Data Availability...
Course Data is Available!

Checking for Degree Requirements...
Degree Requirements are Available!

Cleaning Documents...

Checking for Topics...
Topics are Available!
```

4. Review available topics and enter one or more words into the query prompt.



```
Welcome to the MCS-DS Course Browser!
-------------------------------------

There are 27 courses available to fufill 8 requirements.

Available Topics:
Topic 0: network,course,computer,infrastructure
Topic 1: programming,iot,performance,parallel
Topic 2: text,data,make,language
Topic 3: cloud,compute,service,distribute
Topic 4: data,min,method,course
Topic 5: course,system,learn,application
Topic 6: data,visualization,programming,python
Topic 7: math,hour,prerequisite,graduate
Topic 8: model,include,problem,regression
Topic 9: data,databases,database,model
Topic 10: learn,course,deep,programming
Topic 11: design,software,process,introduce

What subjects are you interested in? [type "quit" to exit]:
```

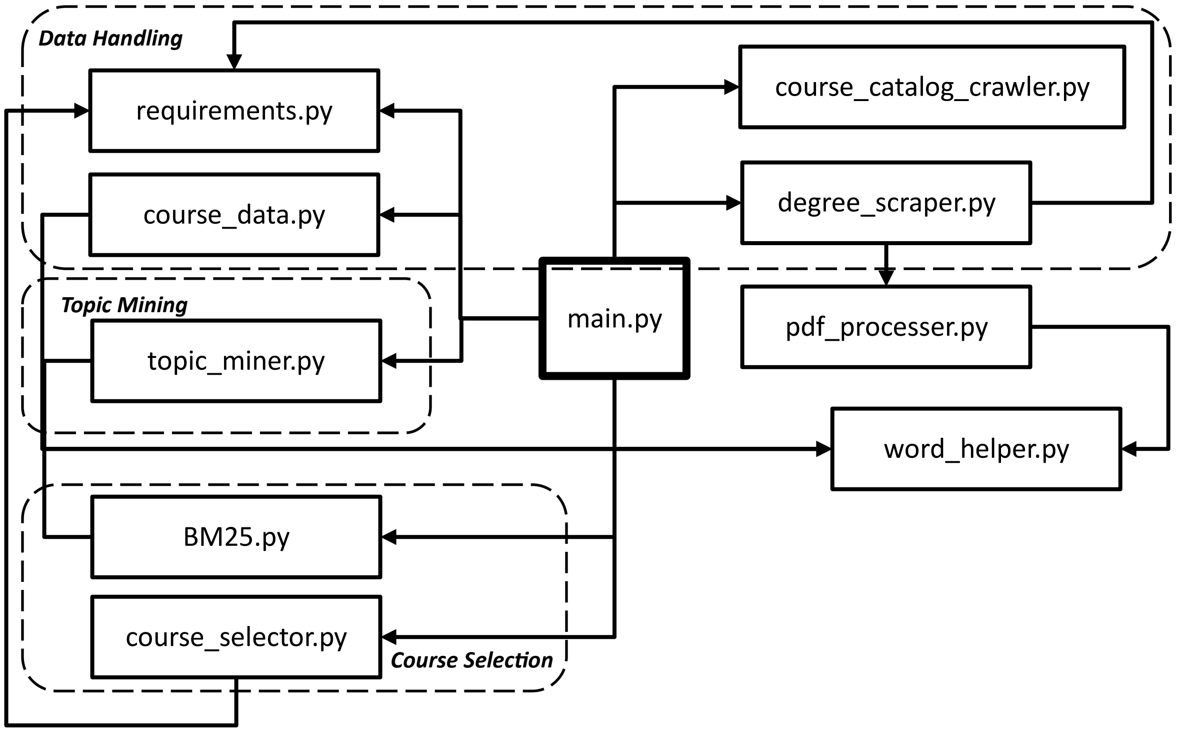
5. Recommended Course List is displayed. The query prompt is repeated in case another selection is desired. If done, please enter “quit” to end the program.



```
What subjects are you interested in? [type "quit" to exit]:  visualization

Recommended Course List:
1) CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
2) CS 416 Data Visualization [Electives]
3) CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
4) CS 598 Deep Learning for Healthcare [Advanced Coursework]
5) CS 410 Text Information Systems [Breadth Coursework-Data Mining]
6) CS 425 Distributed Systems ( Cloud Computing Concepts) [Breadth Coursework-Cloud Computing:]
7) CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
8) CS 598 Foundations of Data Curation [Advanced Coursework]

*WARNING: Please verify any prerequisites are met before using recommended course list!
```

Implementation

In order to implement the MCS-DS Degree Course Selector, a main script, 8 sub-scripts, and a helper script were created. The program has 3 main sections: data handling, topic mining, and course selection.



Data Handling

Data handling is done in two different ways: reading/writing preprocessed data (requirements.py, course_data.py) and reprocessing data from the websites (course_catalog_crawler.py, degree_scraper.py, pdf_processer.py). First, main.py uses requirements.py to grab the requirement specifications from file. Then, it uses course_data.py to grab course titles and descriptions from file. If main.py has any issues loading the preprocessed data, the user is given the option to scrape it from the internet. The degree_scraper.py grabs the requirement specifications and some course data from the degree requirements webpage. Since this courses data is stored in pdf files, the script pdf_processer.py is used to parse out the course description. Then, course_catalog_crawler.py is used to grab the remainder of the course descriptions. If the course exists in both places, the descriptions are combined.

As the final step in data handling, all course descriptions are cleaned. This is done on a word-by-word basis by lowering letters, reducing words to alphabetic characters only, and stemming words to their root to encourage grouping of similar terms.

Courses Discovered

- CS 441 Applied Machine Learning
- CS 445 Computational Photography
- CS 447 Natural Language Processing
- CS 598 Deep Learning for Healthcare
- CS 410 Text Information Systems
- CS 411 Database Systems
- CS 412 Introduction to Data Mining
- CS 416 Data Visualization
- CS 519 Scientific Visualization
- CS 425 Distributed Systems
  (Cloud Computing Concepts)
- CS 435 Cloud Networking
- CS 437 Internet of Things
- CS 498 Cloud Computing Applications
- CS 513 Theory and Practice of Data Cleaning
- CS 598 Foundations of Data Curation
- CS 598 Practical Statistical Learning*
- CS 598 Advanced Bayesian Modeling
- CS 598 Cloud Computing Capstone*
- CS 598 Data Mining Capstone*
- CS 418 Interactive Computer Graphics
- CS 421 Programming Languages and Compilers
- CS 427 Software Engineering I
- CS 450 Numerical Analysis
- CS 463 Computer Security II
- CS 475 Formal Models of Computation
- CS 484 Parallel Programming
- STAT 420 Methods of Applied Statistic

Topic Mining

Topic Mining was performed using the cleaned course titles and course descriptions using the topic_miner.py script. In addition, stopwords from the nltk library were removed from the course descriptions. LDA topic mining was performed searching for the top 4 words in 12 topics.

Topics Discovered

1. network, course, computer, infrastructure
2. programming, iot, performance, parallel
3. text, data, make, language
4. cloud, compute, service, distribute
5. data, min, method, course
6. course, system, learn, application
7. data, visualization, programming, python
8. math, hour, prerequisite, graduate
9. model, include, problem, regression
10. data, databases, database, model
11. learn, course, deep, programming
12. design, software, process, introduce

Course Selection

After the course data, requirements, and topics have been retrieved, the program is ready. First, the mined topics are presented to the user in a list of available topics. Then, the user is prompted to enter their interests. They can choose to pick topics from the list or create their own query.

The user query is cleaned using the same method as the data was cleaned in the data handling section. The cleaned query is sent to the BM25.py script and used to score the relevancy of each course description using the BM25 algorithm with hyperparameters k = 1.0 and b = 0.5. The courses are ranked from highest score to lowest.

The ranked list is sent to the course_selector.py script. The course selector cycles through the ranked list trying to match the course to any unmet degree requirement. A course is only discarded if there are no requirements left for it to satisfy. This continues until all degree requirements are met.

Class Structure

Outlined below are the major class structures of each script, including global variables, data types, and function headers.

course_data.py

```
# global variables
courses_all_filePath = "courses_all.txt"
courses_dso_filePath = "courses_dso.txt"
courses_all_dict = dict()
courses_dso_dict = dict()

# data type, contains information about 1 course
# properties: title (string), description (string)
class Course :

# loads courses into courses_all_dict & courses_dso_dict from files
# using ReadCoursesFromFile(), dictionaries are empty if issue reading files
#   in: N/A
#   out: success (Boolean)
def LoadCourses() :

# reads Courses from file
#   in: filePath
#   out: (dictionary) [key: course title string, value: Course obj]
def ReadCoursesFromFile(file_path) :

# writes all Course dictionaries to file
#   in: N/A
#   out: N/A
def WriteAllCoursesToFile() :

# writes Course to file
#   in: filePath (string), (dictionary) [key: course title string, value: Course obj]
#   out: N/A
def WriteCoursesToFile(file_path, courses_dict) :

# empties out the data structures for re-writing
#   in: N/A
#   out: N/A
def Empty() :

# uses course title to find course in course data
#   in: course_title (string)
#   out: (Course obj)
def FindCourse(course_title) :

# adds course to the correct data structure(s),
# check for overlapping and duplicate courses using FindCourse().
# If match found, merge descriptions.
#   in: course_title (string), (Course obj), dso (Boolean)
#   out: N/A
def AddCourse(course_title, course_description, dso_flag) :
```

```python
# prints course dictionary to console
#   in: (dictionary) [key: course title string, value: Course obj]
#   out: N/A
def PrintCourseDict_Debug(courses_dict) :

# clean document contents
#   in: (dictionary) [key: original course title string, value: Course obj]
#   out: (dictionary) [key: original course title (string), value: clean course description
(list of string)]
def CleanCourses(courses_dict) :
```

requirements.py

```python
# global variables
req_filePath = "requirements.txt"
req_titles = []
req_counts = []
req_courses_dict = dict()

#loads requirements into data objects from file using
#ReadRequirementsFromFile(), data objects empty if issue reading files
#   in: N/A
#   out: success (Boolean)
def LoadRequirements() :

#reads requirement data from file and stores directly in data objects
#   in: N/A
#   out: N/A
def ReadRequirementsFromFile() :

#write requirement data to file
#   in: N/A
#   out: N/A
def WriteRequirementsToFile() :

#empties out the data structures for re-writing
#   in: N/A
#   out: N/A
def Empty() :

#adds requirement to data structure
#   in: req_title (string), course_title (string), num_needed (int), tot_num_needed (int)
#   out: N/A
def AddRequirement(req_title, course_title, num_needed, tot_num_needed = -1) :
```

course_catalog_crawler.py

```python
# global variables
course_catalog_main_url = 'https://siebelschool.illinois.edu'
course_catalog_subpage_url = '/academics/courses'

# scrapes online course catalog for course titles & descriptions
#   in: N/A
#   out: (list of tuple) [course title (string), course description (string)]
def FindCoursesOnline() :
```

degree_scraper.py

```
# global variables
degree_webpage_url = 'https://siebelschool.illinois.edu/academics/graduate/professional-
mcs/online-master-computer-science-data-science'

# translates English number string into integer
def numStrToInt(num_str) :
    numStr_list =
["zero","one","two","three","four","five","six","seven","eight","nine","ten"]
    return numStr_list.index(num_str)

# scrapes online degrees requirements for requirements
# and adds them directly to requirements script, variable r
# AND pulls available course syllabuses
#   in: r (requirements)
#   out: (list of tuple) [course title (string), course description (string)]
def FindRequirements(r) :
```

pdf_processer.py

```
# checks if word is valid according to nltk corpus
#   in: w (string)
#   out: valid_word (Boolean)
def CheckWord(w) :

# tries to remove extra spaces, an issue caused by reading extracting pdf files
#   in: pdf_content (string)
#   out: clean_pdf_content (string)
def CleanUpSpaces(pdf_content) :

# parses the pdf file located at file url for a course description text blob
# and cleans it up using CleanUpSpaces
#   in: file_url (string)
#   out: course_description (string)
def ParsePDFFile_CourseDescription(file_url, debug = False) :
```

topic_miner.py

```
# global variables
topics_filePath = "topics.txt"
topics_list = []

# data type, contains information about 1 topic
# properties: title (string), word_list (list of string)
class Topic :

# loads topics into data objects from file using
# ReadTopicsFromFile(), data objects empty if issue reading files
#   in: N/A
#   out: success (Boolean)
def LoadTopics() :
```

```
# reads topic data from file and stores directly in data objects
#   in: N/A
#   out: N/A
def ReadTopicsFromFile() :

# write topics data to file
#   in: N/A
#   out: N/A
def WriteTopicsToFile() :

# empties out the data structures for re-writing
#   in: N/A
#   out: N/A
def Empty() :
    global topics_list
    topics_list = []

# using LDA, mine topics from list of strings
#   in: text_list (list of string), num_topics (int), n_top_words (int)
#   out: N/A, topics saved internally
def Mine_Topics_lda(text_list, num_topics = 20, n_top_words = 8) :

# mine documents for topics and save results internally
#   in: (dictionary) [key: original course title (string),
#        value: clean course description (list of string)]
#   out: N/A, topics saved internally
def Mine_Topics(clean_docs_dict) :
```

BM25.py

```
# calculate average document length
#   in: (dictionary) [key: original course title (string), value: clean course description
(list of string)]
#   out: avg_doc_len (float)
def calc_avgLen(doc_dict) :

# create a dictionary with the count of every word
#   in: (dictionary) [key: original course title (string), value: clean course description
(list of string)]
#   out: (dictionary) [key: word string, value: doc cnt int]
def calc_word_doc_freq(doc_dict) :

# create a dictionary with the count of every word in the string
#   in: word_list (list of string)
#   out: (dictionary) [key: word string, value: count of word int]
def calc_term_freq(word_list) :

# scores a document based on a query
#   in: query (string), document words (list of string), avg_doc_len (float),
#    num_docs_M (int), word_doc_freq_dict (dictionary) [key: word string, value: doc cnt
int]
#   out: N/A
def BM25 (query, doc_list, avg_doc_len, num_docs_M, word_doc_freq_dict, b, k) :
```

```
# rank documents in collection based on query
#   in: query (string), (dictionary) [key: original course title (string), value: clean
course description (list of string)]
#   out: ranked list of courses by key (list)
def rank_collection(query, clean_docs_dict) :
```

course_selector.py

```
# selects the highest ranked courses from a ranked list
# that meet the degree requirements
#   in: ranked_list (list of string)
#   out: user_course_list (list of string)
def select_from_ranked_list(ranked_list) :
```

word_helper.py

```
# returns a stemmed word from the nltk corpus
# or nothing if word is not easily stemmable
def FindValidStemmedWord(w) :

# clean word by removing non-alphabetical characters and stemming if possible
#   in: word (string)
#   out: clean word (string)
def CleanWord(word, ps = None) :

# clean string of words, optional delimiter specification
#   in: word_str (string), delimiter (string)
#   out: clean words (list of string)
def CleanWordString(word_str, delimiter = ' ') :
```

Verification

The effectiveness of this selector was evaluated using the nDCG@8 score (Normalized Discount Cumulative Gain at 8 documents) of the courses selected for various topic group queries. A human relevancy ranking [0,5] was generated for each course-topic pair using the courses discovered in Data Handling and the topics discovered in Topic Mining.

For each topic query, the nDCG@8 was calculated for both the ranked course list from the BM25 algorithm and the selected course list from the Course Selector algorithm and then recorded in the table below. The UIUC MCS-DS Course Selector is generally quite effective with a majority of the nDCG@8 scores above 0.5. The scores are typically lower in the Selected Courses nDCG@8 vs the BM25 Ranked List nDCG@8 since the degree requirements will force course selection outside of the queried user interests.

Query	*BM25 Ranked List NDCG@8* *Normalized Discount Cumulative Gain @ top 8 courses*	*Selected Courses NDCG@8* *Normalized Discount Cumulative Gain of 8 courses that meet requirements*
"network computer infrastructure"	0.7288	0.5813
"programming iot performance parallel"	0.7701	0.6372
"text data language"	0.5813	0.4671
"cloud compute service distribute"	1.0	0.7759
"data method"	0.7910	0.7910
"system learn application"	0.7982	0.7923
"data visualization programming python"	0.6285	0.6521
"math"	0.4788	0.4393
"model problem regression"	0.7345	0.7756
"data database model"	0.7951	0.8111
"learn deep programming"	0.8390	0.8900
"design software process"	0.7554	0.6826

Recommended Course Lists

Query "network computer infrastructure"

- CS 435 Cloud Networking [Breadth Coursework-Cloud Computing:]
- CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
- CS 498 Cloud Computing Applications [Electives]
- CS 411 Database Systems [Breadth Coursework-Data Mining]
- CS 598 Deep Learning for Healthcare [Advanced Coursework]
- CS 416 Data Visualization [Breadth Coursework-Data Visualization]
- CS 519 Scientific Visualization [Advanced Coursework]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]

Query “programming iot performance parallel”

- CS 484 Parallel Programming [Electives]
- CS 437 Internet of Things [Breadth Coursework-Cloud Computing:]
- CS 598 Deep Learning for Healthcare [Breadth Coursework-Machine Learning]
- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
- CS 412 Introduction to Data Mining [Breadth Coursework-Data Mining]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
- CS 598 Foundations of Data Curation [Advanced Coursework]
- CS 598 Practical Statistical Learning* [Advanced Coursework]

Query “text data language”

- CS 410 Text Information Systems [Breadth Coursework-Data Mining]
- CS 421 Programming Languages and Compilers [Electives]
- CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
- CS 425 Distributed Systems (Cloud Computing Concepts) [Breadth Coursework-Cloud Computing:]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
- CS 416 Data Visualization [Breadth Coursework-Data Visualization]
- CS 598 Foundations of Data Curation [Advanced Coursework]
- CS 598 Data Mining Capstone* [Advanced Coursework]

Query “cloud compute service distribute”

- CS 498 Cloud Computing Applications [Breadth Coursework-Cloud Computing:]
- CS 437 Internet of Things [Electives]
- CS 598 Cloud Computing Capstone* [Advanced Coursework]
- CS 411 Database Systems [Breadth Coursework-Data Mining]
- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
- CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
- CS 598 Deep Learning for Healthcare [Advanced Coursework]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]

Query “data method”

- CS 412 Introduction to Data Mining [Breadth Coursework-Data Mining]
- CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
- CS 598 Deep Learning for Healthcare [Advanced Coursework]
- CS 598 Advanced Bayesian Modeling [Advanced Coursework]
- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
- CS 416 Data Visualization [Electives]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
- CS 498 Cloud Computing Applications [Breadth Coursework-Cloud Computing:]

Query “system learn application”

- CS 412 Introduction to Data Mining [Breadth Coursework-Data Mining]
- CS 498 Cloud Computing Applications [Breadth Coursework-Cloud Computing:]
- CS 410 Text Information Systems [Electives]
- CS 416 Data Visualization [Breadth Coursework-Data Visualization]
- CS 598 Deep Learning for Healthcare [Breadth Coursework-Machine Learning]
- CS 598 Cloud Computing Capstone* [Advanced Coursework]
- CS 598 Practical Statistical Learning* [Advanced Coursework]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]

Query “data visualization programming python”

- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
- CS 416 Data Visualization [Electives]
- CS 435 Cloud Networking [Breadth Coursework-Cloud Computing:]
- CS 598 Deep Learning for Healthcare [Breadth Coursework-Machine Learning]
- CS 412 Introduction to Data Mining [Breadth Coursework-Data Mining]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
- CS 598 Foundations of Data Curation [Advanced Coursework]
- CS 598 Data Mining Capstone* [Advanced Coursework]

Query “math”

- CS 447 Natural Language Processing [Breadth Coursework-Machine Learning]
- CS 418 Interactive Computer Graphics [Electives]
- CS 598 Deep Learning for Healthcare [Advanced Coursework]
- CS 410 Text Information Systems [Breadth Coursework-Data Mining]
- CS 416 Data Visualization [Breadth Coursework-Data Visualization]
- CS 519 Scientific Visualization [Advanced Coursework]
- CS 425 Distributed Systems (Cloud Computing Concepts) [Breadth Coursework-Cloud Computing:]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]

Query “model problem regression”

- CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
- CS 598 Advanced Bayesian Modeling [Advanced Coursework]
- STAT 420 Methods of Applied Statistics [Electives]
- CS 598 Deep Learning for Healthcare [Advanced Coursework]
- CS 410 Text Information Systems [Breadth Coursework-Data Mining]
- CS 598 Cloud Computing Capstone* [Advanced Coursework]
- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
- CS 498 Cloud Computing Applications [Breadth Coursework-Cloud Computing:]

Query “data database model”

- CS 411 Database Systems [Breadth Coursework-Data Mining]
- CS 498 Cloud Computing Applications [Breadth Coursework-Cloud Computing:]
- CS 598 Deep Learning for Healthcare [Breadth Coursework-Machine Learning]
- CS 598 Advanced Bayesian Modeling [Advanced Coursework]
- STAT 420 Methods of Applied Statistics [Electives]
- CS 598 Practical Statistical Learning* [Advanced Coursework]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]

Query “learn deep programming”

- CS 598 Deep Learning for Healthcare [Breadth Coursework-Machine Learning]
- CS 437 Internet of Things [Breadth Coursework-Cloud Computing:]
- CS 519 Scientific Visualization [Breadth Coursework-Data Visualization]
- CS 441 Applied Machine Learning [Electives]
- CS 412 Introduction to Data Mining [Breadth Coursework-Data Mining]
- CS 598 Cloud Computing Capstone* [Advanced Coursework]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]
- CS 598 Foundations of Data Curation [Advanced Coursework]

Query “design software process”

- CS 427 Software Engineering I [Electives]
- CS 498 Cloud Computing Applications [Breadth Coursework-Cloud Computing:]
- CS 410 Text Information Systems [Breadth Coursework-Data Mining]
- CS 416 Data Visualization [Breadth Coursework-Data Visualization]
- CS 441 Applied Machine Learning [Breadth Coursework-Machine Learning]
- CS 598 Deep Learning for Healthcare [Advanced Coursework]
- CS 519 Scientific Visualization [Advanced Coursework]
- CS 513 Theory and Practice of Data Cleaning [Advanced Coursework]