Note: All Python code used in this story is copied at the bottom of this document and available on Github. My final data spreadsheet is available to view at this link, with minor explanatory notes added.

In late 2022, ChatGPT was released to the public, disrupting the paradigms of university education with its ability to generate believable written essays and assignments. News outlets covered the way universities and university instructors were responding to this new technology — from those who wanted to ban it outright to others who embraced it into their pedagogy.

By the fall of 2023, about a year after ChatGPT's launch, I became curious about what, if anything, had actually changed at the university level to respond to generative AI and ChatGPT. Certainly, students were using it to complete assignments and essays, but I was curious what approach professors were taking, after a few semesters to figure out how to respond. I also wanted to understand how things might vary across departments/faculties. I felt a data-based approach would be the best way to understand the overall pattern.

To understand how generative AI/ChatGPT use has affected individual courses, I decided to focus on course syllabi, where all course policies must be recorded and which would therefore include any regulations concerning ChatGPT. I first copied a list of fall semester courses (September to December 2023) and full year courses (September 2023 to April 2024) across all undergraduate programs at Western University using a class-schedule tool for students.

Syllabi were downloaded from course pages and department websites, as necessary, to create the set of syllabi used in the analysis. For all missing syllabi, professors and/or department staff were emailed to request a copy of the course's syllabus.

I was able to obtain syllabi for approximately two-thirds of the approximately 1,000 courses offered that met the above criteria. I checked the number of syllabi I had obtained per faculty against the number of courses offered by each faculty to ensure that the number of courses I had for each faculty was acceptable, which meant 30% of each faculty's syllabi at a minimum.

Once I had downloaded each syllabus (as .pdf files), I then used Python and the external PyPDF2 library, as well as the built-in os and csv libraries, to search each .pdf file for a set of keywords, which was created based on an initial scan of 50 syllabi and reading articles on generative AI in higher education.

My Python script checked each of the syllabi to see which, if any, keywords it contained. It then wrote the name of the course and the keywords to a .csv file. In addition to the keywords, I also included a column for the syllabus character count (as recorded by PyPDF2) in my final .csv file. This step ensured that if the PyPDF2 scanner failed to accurately check a syllabus (e.g. if it scanned through a small number of characters, from 0 to ~1000), I would be able to catch this error and manually scan that syllabus for keywords.

Extracting keywords using Python allowed me to narrow down the approximately 650 syllabi I started out with to approximately 200 that contained a keyword. For the syllabi that contained a keyword(s), I manually searched for the keyword(s) in the .pdf file and copied the section of the syllabus that discussed the keyword(s) into a spreadsheet.

In the spreadsheet, I completed my data analysis of the ~200 syllabi containing a keyword(s) manually (using built-in functions in Google Sheets to aggregate the results). I recorded whether a syllabus contained any of a list of characteristics in its generative AI policy (e.g. citing generative AI if used, use of generative AI detection tools, etc.). I also excluded syllabi that discussed generative AI as part of the course curriculum (e.g. a computer science course about artificial intelligence) but did not have a discussion of generative AI use in coursework. Lastly, I scored each syllabus as positive, neutral or negative towards the use of generative AI.

To ensure consistency in my scoring, I completed the above analysis twice, one week apart, and resolved any discrepancies (which ended up being applicable in only two instances) by consulting three other co-workers.

After completing my analysis, I used basic functions in Google Sheets to aggregate the results for visualization. I used Flourish Studio to create the interactive visualizations used in the article, including pie charts, a scatterplot, bar chart, and a proportional stacked bar chart. I also collaborated with graphics staff at my student newspaper to generate visuals highlighting unique ways professors were using ChatGPT/generative AI in their assessments.

The results of my analysis informed the sources I decided to speak to. I spoke to professors from a variety of faculties and departments who could contextualize the data. My sources helped to answer the “why” part of my story. For example, why did certain departments mention ChatGPT more than others? I also spoke to administrators to get a high-level perspective, and the ombudsperson to understand the consequences (e.g. ChatGPT and reports of plagiarism) of generative AI over the last year.

In the future, I would like to experiment with natural language processing tools to create a sentiment classifier. This might expedite the process of completing the analysis of whether instructors felt positively, negatively, or were neutral towards generative AI and ChatGPT. However, I feel there is a small risk of bias and inaccuracies that could be introduced. Additionally, I’d like to figure out ways to address notable limitations in my work such as not obtaining a complete sample of syllabi.

Code to Scan Documents for Keywords:

```
from PyPDF2 import PdfReader
from PyPDF2 import errors
import csv
import os
```

```
# Keywords
words = ['ai', 'chatgpt', 'llm', 'artificial intelligence',
'language model', 'gpt', 'generative ai', 'midjourney',
         'bard', 'dall-e']

# Get faculties
folder_name = 'syllabi'
faculties = os.listdir(os.path.join(os.getcwd(), folder_name))
faculties.remove(".DS_Store")

# Get departments inside the faculty folder (list)
departments = []
for faculty in faculties:
    faculty_departments = os.listdir(os.path.join(os.getcwd(),
folder_name, faculty))
    for department in faculty_departments:
        if department != ".DS_Store":
            departments.append(f'{faculty}/{department}')

# Get file names for all syllabi within department folder (list)
syllabi = []
for department in departments:
    department_syllabi = os.listdir(os.path.join(os.getcwd(),
folder_name, department))
    for syllabus in department_syllabi:
        if syllabus != ".DS_Store":
            syllabi.append(f'{department}/{syllabus}')


# Main method to check each syllabus and write output to csv
def main():
    for syllabus_path in syllabi:
        try:
            text = scan_syllabus(os.path.join(os.getcwd(),
folder_name, syllabus_path))
            keywords = find_keywords(text)
            write_to_csv(syllabus_path, keywords, len(text))
        except errors.DependencyError:
            print('Dependency Error!! (1)\n', syllabus_path)
        except errors.PdfReadError:
            print('This PDF is not able to be read: ',
syllabus_path)
```

```
# Function to scan syllabus using PdfReader objects
def scan_syllabus(syllabus_text):
    text = ""
    reader = PdfReader(syllabus_text)
    for page in reader.pages:
        try:
            text = text + page.extract_text().strip()
        except errors.DependencyError:
            print('Dependency Error 2\n', syllabus_text)
    return text


# Function to find keywords in text passed from the syllabus pdf
def find_keywords(text):
    keywords = []
    text = text.lower()  # so the scan is NOT case-sensitive
    for word in words:
        if " " + word + " " in text:  # to avoid, for e.g., "said"
            keywords.append(word)
        elif " " + word + "." in text:
            keywords.append(word)
        elif " " + word + "," in text:
            keywords.append(word)
    return keywords


# Function to write output to csv files
def write_to_csv(syllabus_path, keywords, num_chars):
    with open('ai_data.csv', 'a') as csv_file:
        writer = csv.writer(csv_file)
        writer.writerow([syllabus_path, keywords, num_chars])
    print(f'Wrote {syllabus_path} to CSV')


main()
```

Code to Count Number of Syllabi With a Generative AI Policy Per Faculty:

```
import csv

ai_values = {}

with open('ai_data.csv', 'r') as f:
```

```
    reader = csv.reader(f)
    for row in reader:
        faculty = row[0].split('/')[0]
        row[1] = row[1].strip('[]')
        if faculty not in ai_values:
            if row[1]:  # Empty string is falsy
                ai_values[faculty] = [1, 1]  # 1 value, 0 mention
AI
            else:
                ai_values[faculty] = [1, 0]  # 1 value, 0 mention
AI
        else:
            if row[1]:
                ai_values[faculty][1] += 1  # Add 1 value for
mention AI
            ai_values[faculty][0] += 1


print(ai_values)
```