Nuclear Safety Culture:
Automation Implementation to Diablo Canyon Safety Reports

ISE 370 Human Factors in Work Design
Department of Industrial and Systems Engineering

BY:

Iris Gordo

Shelby Wu

December 9th, 2022

Abstract

Every year, mass amounts of nuclear safety reports are written and analyzed to dissect the errors of human factors and prevent future mistakes from happening again. Identifying these traits are crucial when learning from the errors caused by human factors, specifically in the nuclear field. To overcome this issue, a discussion is presented on how a Python program can identify what areas of nuclear safety reports relate to at least one of the safety traits provided by the Institute of Nuclear Power Operations's (INPO) Guide of Safety Culture Traits. The remaining sections discuss how the foundation of our research and analysis is based on the INPO's guide for safety culture traits and Diablo Canyon's Nuclear Safety Report. The results of the program are also presented in which it represents the safety culture traits identified in Diablo Canyon's Nuclear Safety Report. Finally, a discussion is presented on how our Python program could have improved for a faster and more efficient process if given more time to work on the project.

Introduction

With safety reports ranging from five to hundreds of pages, examining and identifying healthy nuclear safety culture traits can take weeks and be a tedious process. Furthermore, many major incidents could have been prevented if quicker understanding and implementation took precedence. However, the labor of reading through countless reports is currently an inefficient system within society. For example, two USC students compared the annual safety report of the Diablo Canyon Nuclear Power Plant to a document describing the safety standards for power plants to illustrate that Diablo Canyon did not follow the safety traits described as well as to identify and explain how they can improve. However, it took months for the students to thoroughly review the report and meet their goal since the safety report was more than 9000 pages long. To combat such issues, implementing and utilizing today's technological advancements such as programming could create a more efficient and productive system in identifying healthy nuclear safety traits.

In the article "Traits of a Healthy Nuclear Safety Culture" by INPO, ten key healthy components to nuclear safety are identified. The components are personal accountability, questioning attitude, effective safety communication, leadership safety values and actions, decision making, respectful work environment, continuous learning, problem identification and resolution, environment for raising concerns, and work process. After establishing an understanding of what each of these traits entail, Python programming will be utilized to scan through nuclear safety reports and pinpoint the areas in which healthy traits appeared. This process would amplify the amount of reports read per day and set a standard for implementation that is faster compared to current processes. Utilizing the technological advancements society has pushed towards, a greater precedence can take place concerning the human factors of nuclear safety.

Background

As mentioned above, this project takes the ten traits from "Traits of a Healthy Nuclear Safety Culture" by INPO and finds these traits within the Diablo Canyon Safety Reports to pinpoint areas of improvement and change. Below is a list of the ten traits and what each of them mean in terms of safety culture.

1. Personal Accountability: All individuals take personal responsibility for safety.

2. Questioning Attitude: Individuals avoid complacency and continuously challenge existing conditions and activities in order to identify discrepancies that might result in error or inappropriate action.
3. Effective Safety Communication: Communications maintain a focus on safety.
4. Leadership Safety Values and Actions: Leaders demonstrate a commitment to safety in their decisions and behaviors.
5. Decision-Making: Decisions that support or affect nuclear safety are systematic, rigorous, and thorough.
6. Respectful Work Environment: Trust and respect permeate the organization.
7. Continuous Learning: Opportunities to learn about ways to ensure safety are sought out and implemented.
8. Problem Identification and Resolution: Issues potentially impacting safety are promptly identified, fully evaluated, and promptly addressed and corrected commensurate with their significance.
9. Environment for Raising Concerns: A safety- conscious work environment (SCWE) is maintained where personnel feel free to raise safety concerns without fear of retaliation, intimidation, harassment, or discrimination.
10. Work Processes: The process of planning and controlling work activities is implemented so that safety is maintained.

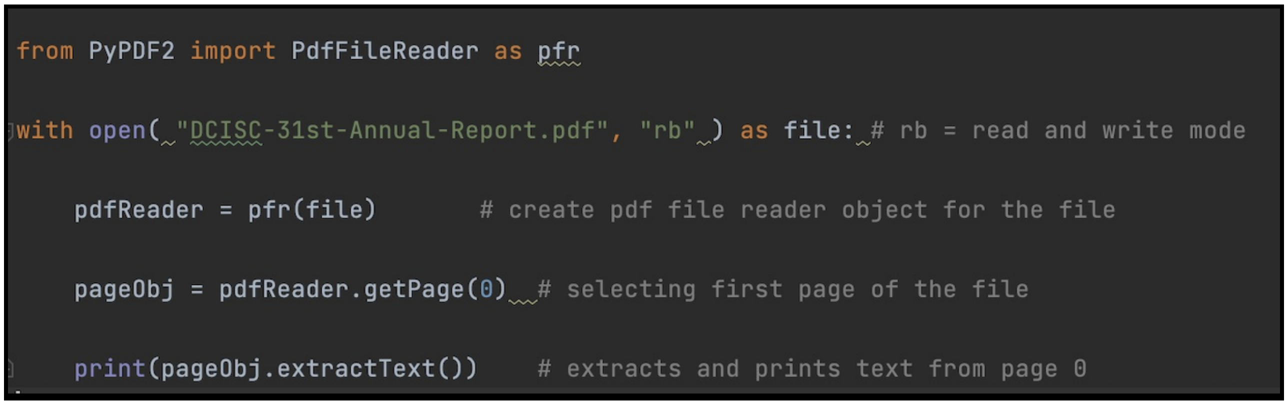
After establishing an understanding of these traits, this paper aims to identify these traits within safety reports through research, key words, and phrase matching to create an automated and efficient system.

Literature Review

We decided to seek guidance from Nathan Greenfield, one of our professors in the Industrial and Systems Engineering department at USC, since he teaches Python. He recommended that we split the project into 2 parts: 1) Processing Files and 2) Context Matching. For part 1, he emphasized that it is important to determine what type of files will be processed in our program. Afterwards, we need to research how we're going to process that file and extract words from the report to eventually accomplish part 2 which is matching certain words in Diablo Canyon's safety report with INPO's nuclear safety traits.

Since the majority of nuclear safety reports are PDF files, we decided that we would focus on importing PDF files for our project. A common way of importing PDF files was by downloading the following Python libraries: pyPDF, PyPDF2, and PyPDF 4. Any of these libraries allow us to achieve our goal but we decided to use PyPDF2 since it is "lightweight, fast, and well-documented" according to an article written by Nanonets. After we decided what type of file we want to import and what Python library we're going to use, we discovered 2 ways of downloading and extracting words from the report provided by *Like Geeks/GeeksForGeeks* and *Medium*.

Like Geeks recommended that we save the file in the same directory where the Python program file is saved so that we can access the safety report via the Python program. Next, we need to extract the text from the pages for processing by opening the PDF file in 'rb' mode or read and binary mode. In this way, the Python program reads the file in a binary format to recognize all types of data besides text such as images. As a result, processing is easier and more accurate. Then, we must create an object for the file since it is easier to use Python functions with objects. Finally, we can extract text from each page using extract(Text) to convert all the text from the PDF file to a string—a collection of alphabets, words or other characters. Using these steps, the code appears as the following:



```
from PyPDF2 import PdfFileReader as pfr

with open( "DCISC-31st-Annual-Report.pdf", "rb" ) as file: # rb = read and write mode

    pdfReader = pfr(file)       # create pdf file reader object for the file

    pageObj = pdfReader.getPage(0)  # selecting first page of the file

    print(pageObj.extractText())    # extracts and prints text from page 0
```

Medium, on the other hand, has a different approach to process files. They first recommend converting the PDF file to txt format and reading the data similar to the *Like Geeks* and *GeekForGeeks* method. Next, they want us to use the .findall() function to extract words that appear in a pattern. Afterwards, we need to create a list of those extracted keywords and save it to a DataFrame for reference. Then, they recommend using Term Frequency-Inverse Document Frequency to calculate a weight for each word to determine how relevant each word is. Finally, we are to save the results to a DataFrame and use the .sort_values() function to arrange the keywords in order. In this step, the results will be saved and sorted in the DataFrame for organization.

After assessing both methods, we decided to follow the process by *Like Geeks* and *GeeksForGeeks* since *Medium*'s process is more optimal for small files. Safety reports, especially Diablo Canyon's safety report, have hundreds of pages. Therefore, it is necessary to separate the document into parts for the convenience and efficiency of the program.

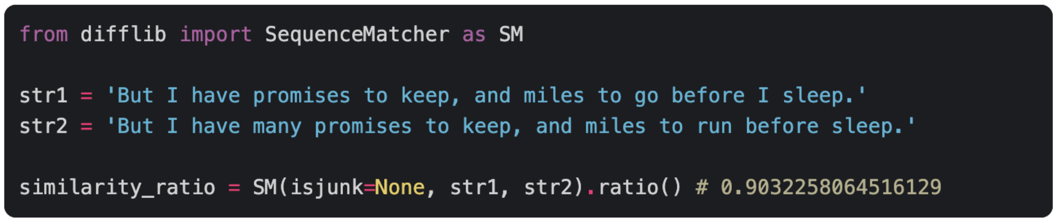
The second part of the project is the context matching phase to pair keywords with INPO's safety culture traits. Since the goal for our program is to associate certain keywords and phrases to the 10 safety culture traits, we needed to research algorithms to do so. Algorithms such as Soundex, New York State Identification and Intelligence System Phonetic algorithm (NYSIIS), or "Fuzzy" matching all closely match to reach our goal for the project.

To summarize, Soundex uses the check_context_char function to check the ambiguity and ensure they are contextually the same name. NYSIIS generates a phonetic code for each input name. More specifically, it indexes the words we input by their pronounciation and returns a phonetic code of that word. Phonetic codes are similar to morse code, but instead, each letter in the alphabet is assigned a word. Lastly, Fuzzy matching uses the function name_matching to return the average ratio of two strings based on character. In other words, it checks how similar two words are based on their characters. Ultimately, we chose to use "Fuzzy" matching due to having the complexity and functions of the other algorithms.

Part of "Fuzzy" Matching is "Fuzzy" Search is the process of finding strings that approximately match a given string. In this case, it will take the words from the nuclear safety reports and match it with words we found that are commonly stated in the report that are associated with at least one of INPO's safety traits. Within the "Fuzzy" Matching algorithm, there are 3 different approaches: 1) SequenceMatcher from difflib, 2) Levenshtein distance, and 3) Fuzzywuzzy.

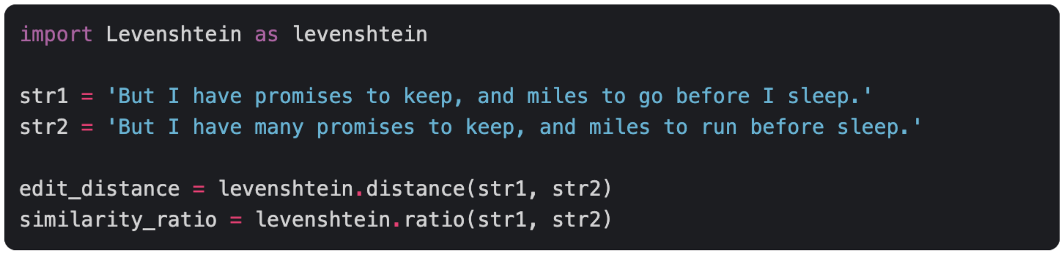
1) The SequenceMatcher approach uses Python's standard library and Ratcliff/Obershelp string matching algorithm to calculate the similarity metric between two strings. To calculate how similar two strings are, the program takes twice the number of matching (overlapping) characters between the two strings divided by the total number of characters in the two strings and returns the ratio. For instance, if the ratio is 0.901, the

similarity between both strings is about 90.1%. The following image is an example of how the code would appear if we chose this method:



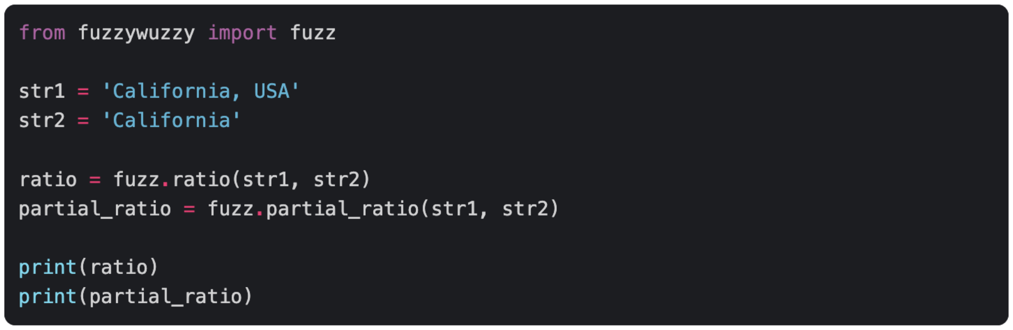
```python
from difflib import SequenceMatcher as SM

str1 = 'But I have promises to keep, and miles to go before I sleep.'
str2 = 'But I have many promises to keep, and miles to run before sleep.'

similarity_ratio = SM(isjunk=None, str1, str2).ratio() # 0.9032258064516129
```

2) The Levenshtein distance between two strings is the number of deletions, insertions and substitutions needed to transform one string into another. In other words, it counts the number of changes that need to be made between two words or phrases. The following is an example of the code:

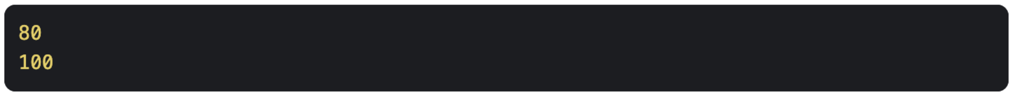


```python
import Levenshtein as levenshtein

str1 = 'But I have promises to keep, and miles to go before I sleep.'
str2 = 'But I have many promises to keep, and miles to run before sleep.'

edit_distance = levenshtein.distance(str1, str2)
similarity_ratio = levenshtein.ratio(str1, str2)
```

3) Finally, Fuzzywuzzy is a more feature-rich library for computing string similarity and performing fuzzy string matching in Python. First, the algorithm calculates a ratio that describes how similar each string is. It can also calculate the partial ratio which is the “ratio similarity measure between the shorter string and every substring of length *m* of the longer string, and returns the maximum of those similarity measures” according to an article on *Medium*. The following is an example of how the Fuzzywuzzy code would generally look like:



```python
from fuzzywuzzy import fuzz

str1 = 'California, USA'
str2 = 'California'

ratio = fuzz.ratio(str1, str2)
partial_ratio = fuzz.partial_ratio(str1, str2)

print(ratio)
print(partial_ratio)
```



```
80
100
```

After analyzing each approach, we decided to use Fuzzywuzzy since it is the most complex and incorporates aspects from both the SequenceMatcher and Levenshtein distance. For a faster and more efficient analysis of the safety report, we will be using loops and conditions in our program. Loops are essentially code that repeats until the conditions are satisfied. Generally, we want our code to follow these steps:

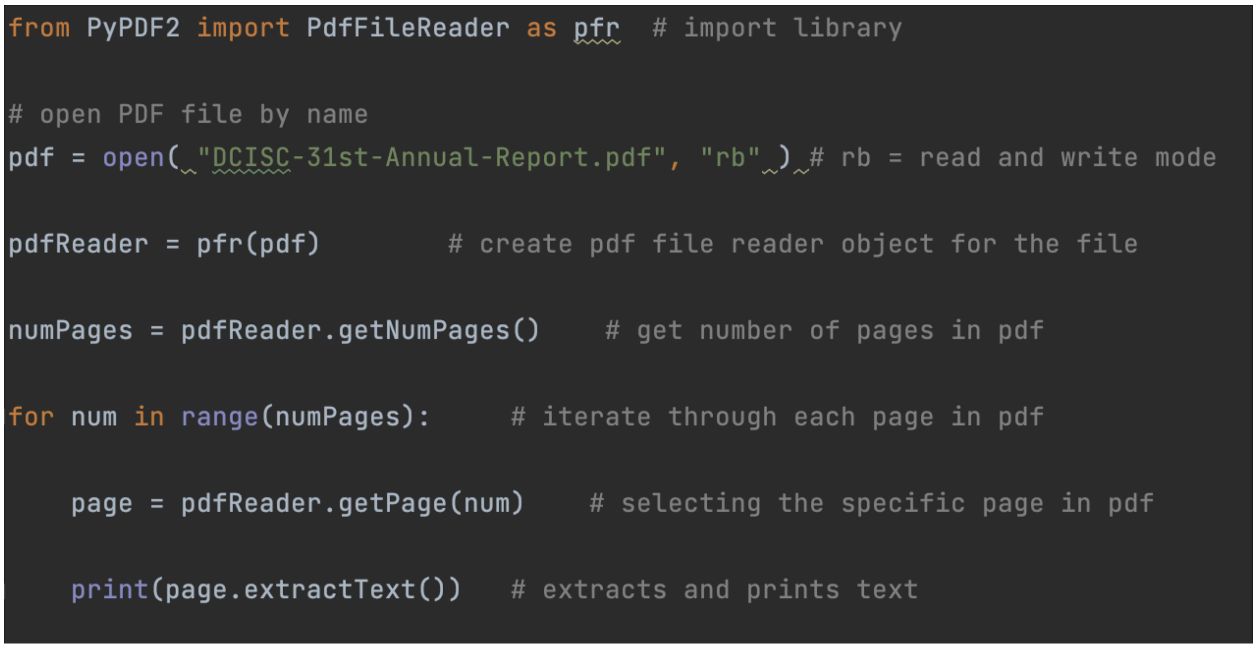
1. Use the ratio function to find areas where the ratio of the safety report compared to each of the safety traits by using key words / phrases of the traits.
    a. If else conditioning will disregard anything below 70% matching. If the ratio is below 70%, the data will go through the handing out of order function (described in step 2) to account for swapped words. We decided to use 70% as our minimum ratio to account for partial matching as some ideas may not be "word for word" within the report.
2. Use the handling out of order function for data that returned below 70% ratio since the order of words can be switched sometimes. However, we will still check for 100% match accuracy for each specific trait.

Methodology

To accomplish the goal of automation, our project employed the research above to accomplish the two tasks: 1) import pdfs of Diablo Canyon Safety Reports into Python PyCharm and 2) scan through the text from each page and use "fuzzy" matching to identify the ten traits within the report.

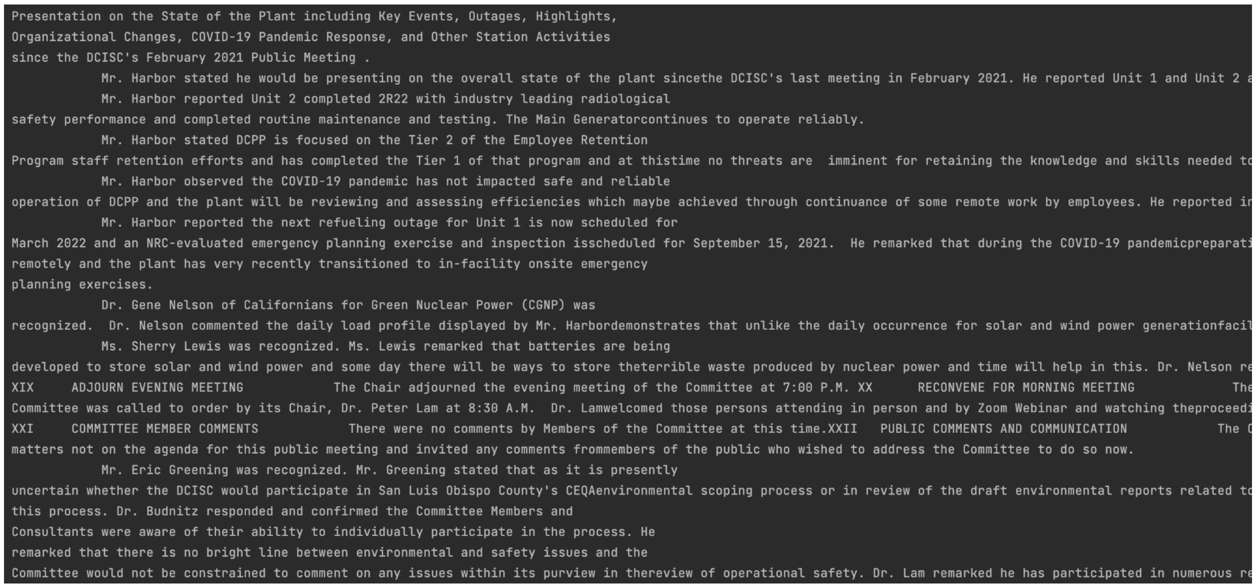
To import the pdf file, our project utilized the PyPDF2 library to import the Diablo Canyon Safety Files. Below is a picture of the program used for testing the download process of the file into PyCharm and the results of printing the first page.

Code:



```python
from PyPDF2 import PdfFileReader as pfr  # import library

# open PDF file by name
pdf = open( "DCISC-31st-Annual-Report.pdf", "rb" ) # rb = read and write mode

pdfReader = pfr(pdf)        # create pdf file reader object for the file

numPages = pdfReader.getNumPages()    # get number of pages in pdf

for num in range(numPages):     # iterate through each page in pdf

    page = pdfReader.getPage(num)    # selecting the specific page in pdf

    print(page.extractText())   # extracts and prints text
```

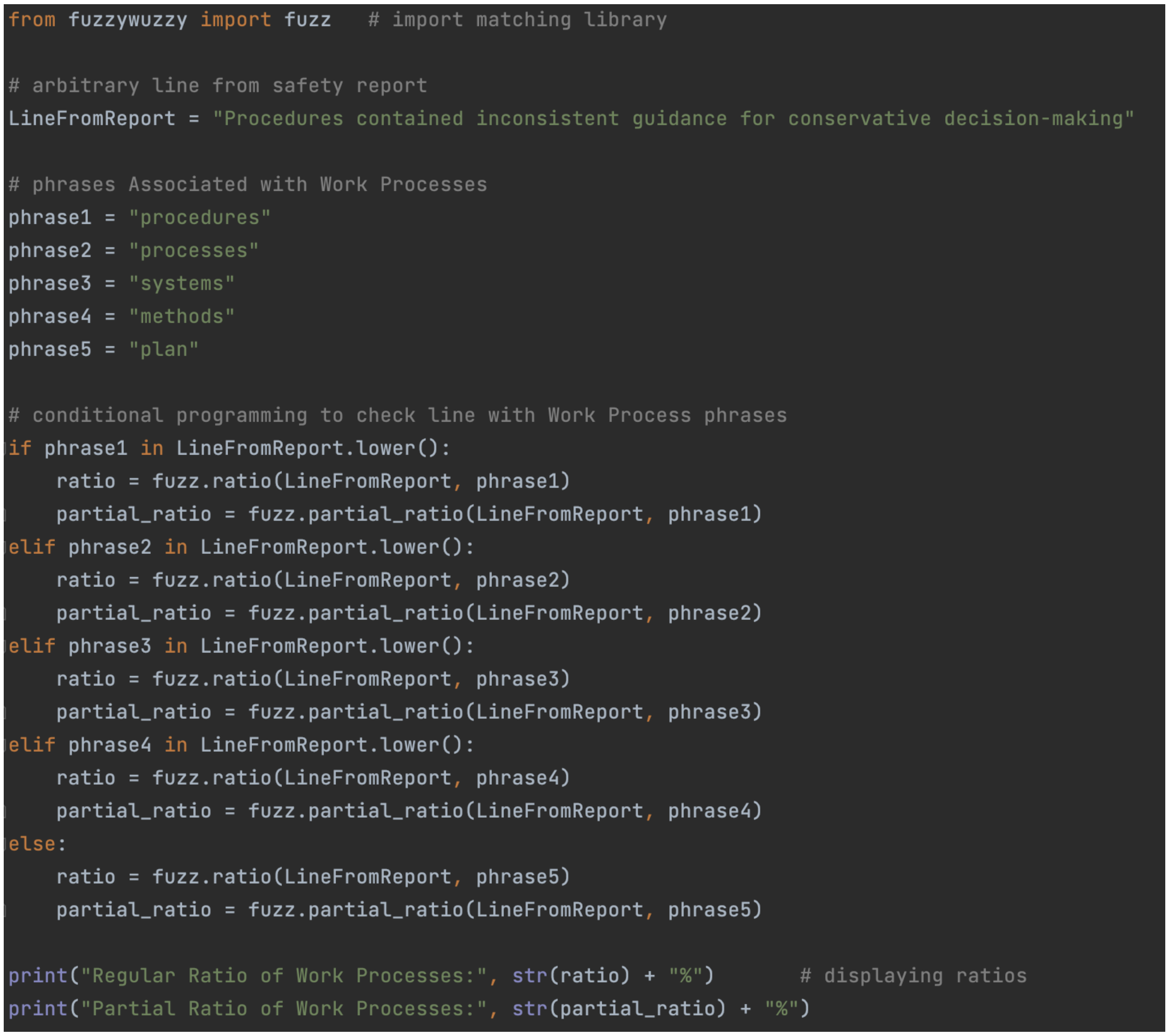
Results:



```
Presentation on the State of the Plant including Key Events, Outages, Highlights,
Organizational Changes, COVID-19 Pandemic Response, and Other Station Activities
since the DCISC's February 2021 Public Meeting .
            Mr. Harbor stated he would be presenting on the overall state of the plant sincethe DCISC's last meeting in February 2021. He reported Unit 1 and Unit 2 a
            Mr. Harbor reported Unit 2 completed 2R22 with industry leading radiological
safety performance and completed routine maintenance and testing. The Main Generatorcontinues to operate reliably.
            Mr. Harbor stated DCPP is focused on the Tier 2 of the Employee Retention
Program staff retention efforts and has completed the Tier 1 of that program and at thistime no threats are  imminent for retaining the knowledge and skills needed to
            Mr. Harbor observed the COVID-19 pandemic has not impacted safe and reliable
operation of DCPP and the plant will be reviewing and assessing efficiencies which maybe achieved through continuance of some remote work by employees. He reported in
            Mr. Harbor reported the next refueling outage for Unit 1 is now scheduled for
March 2022 and an NRC-evaluated emergency planning exercise and inspection isscheduled for September 15, 2021.  He remarked that during the COVID-19 pandemicpreparati
remotely and the plant has very recently transitioned to in-facility onsite emergency
planning exercises.
            Dr. Gene Nelson of Californians for Green Nuclear Power (CGNP) was
recognized.  Dr. Nelson commented the daily load profile displayed by Mr. Harbordemonstrates that unlike the daily occurrence for solar and wind power generationfacil
            Ms. Sherry Lewis was recognized. Ms. Lewis remarked that batteries are being
developed to store solar and wind power and some day there will be ways to store theterrible waste produced by nuclear power and time will help in this. Dr. Nelson re
XIX     ADJOURN EVENING MEETING            The Chair adjourned the evening meeting of the Committee at 7:00 P.M. XX      RECONVENE FOR MORNING MEETING            The
Committee was called to order by its Chair, Dr. Peter Lam at 8:30 A.M.  Dr. Lamwelcomed those persons attending in person and by Zoom Webinar and watching theproceedi
XXI     COMMITTEE MEMBER COMMENTS            There were no comments by Members of the Committee at this time.XXII   PUBLIC COMMENTS AND COMMUNICATION            The C
matters not on the agenda for this public meeting and invited any comments frommembers of the public who wished to address the Committee to do so now.
            Mr. Eric Greening was recognized. Mr. Greening stated that as it is presently
uncertain whether the DCISC would participate in San Luis Obispo County's CEQAenvironmental scoping process or in review of the draft environmental reports related to
this process. Dr. Budnitz responded and confirmed the Committee Members and
Consultants were aware of their ability to individually participate in the process. He
remarked that there is no bright line between environmental and safety issues and the
Committee would not be constrained to comment on any issues within its purview in thereview of operational safety. Dr. Lam remarked he has participated in numerous re
```

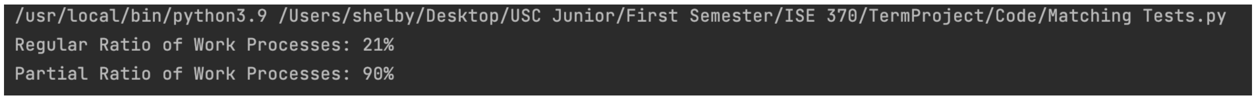
After testing out the download code for running the pdf in PyCharm, the next step was to implement matching and comparison of this text to the safety trait's characteristics. Before attempting to combine both steps, an initial simplistic test was done to try out the "fuzzy matching" code first. To do so, the library, fuzzywuzzy, was imported to match words together and execute ratios of comparison. A variable was established to hold a line from the report and the program checked this line with phrases that represented Work Processes (one of the safety traits). Using conditional programming, the code was able to scan and find ratios of regular and partial matching for this line of text. This and the results of the ratios are shown below.

Code:



```python
from fuzzywuzzy import fuzz   # import matching library

# arbitrary line from safety report
LineFromReport = "Procedures contained inconsistent guidance for conservative decision-making"

# phrases Associated with Work Processes
phrase1 = "procedures"
phrase2 = "processes"
phrase3 = "systems"
phrase4 = "methods"
phrase5 = "plan"

# conditional programming to check line with Work Process phrases
if phrase1 in LineFromReport.lower():
    ratio = fuzz.ratio(LineFromReport, phrase1)
    partial_ratio = fuzz.partial_ratio(LineFromReport, phrase1)
elif phrase2 in LineFromReport.lower():
    ratio = fuzz.ratio(LineFromReport, phrase2)
    partial_ratio = fuzz.partial_ratio(LineFromReport, phrase2)
elif phrase3 in LineFromReport.lower():
    ratio = fuzz.ratio(LineFromReport, phrase3)
    partial_ratio = fuzz.partial_ratio(LineFromReport, phrase3)
elif phrase4 in LineFromReport.lower():
    ratio = fuzz.ratio(LineFromReport, phrase4)
    partial_ratio = fuzz.partial_ratio(LineFromReport, phrase4)
else:
    ratio = fuzz.ratio(LineFromReport, phrase5)
    partial_ratio = fuzz.partial_ratio(LineFromReport, phrase5)

print("Regular Ratio of Work Processes:", str(ratio) + "%")       # displaying ratios
print("Partial Ratio of Work Processes:", str(partial_ratio) + "%")
```

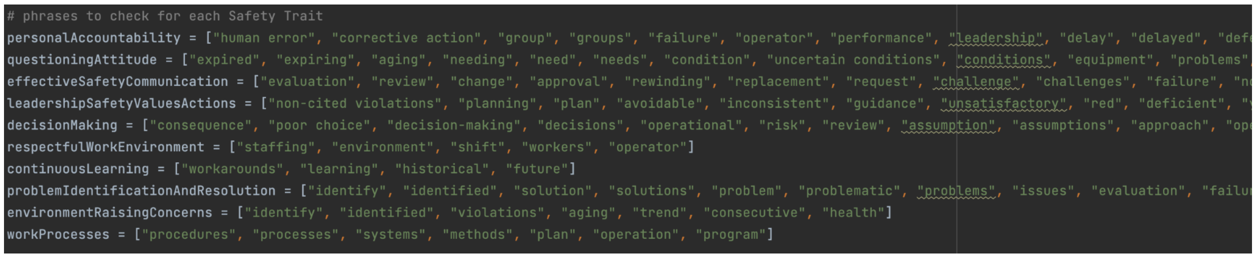
Results:



```
/usr/local/bin/python3.9 /Users/shelby/Desktop/USC Junior/First Semester/ISE 370/TermProject/Code/Matching Tests.py
Regular Ratio of Work Processes: 21%
Partial Ratio of Work Processes: 90%
```

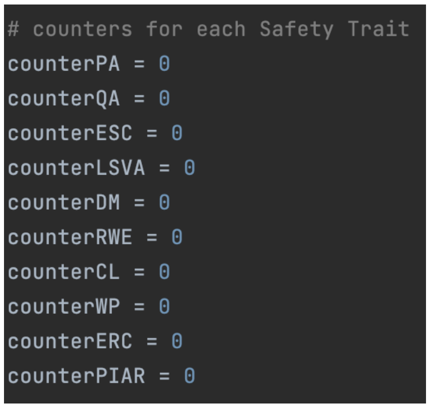
Once both parts were completed and working, combining the two aspects needed some additional work and rearranging. To establish the safety traits, each trait has a list of words and phrases that were utilized based on research. This research included reading the reports, reading “Traits of a Healthy Nuclear Safety Culture” by INPO, and analyzing a previous term paper. After collecting these words and phrases, they will be compared to the text in each page which will determine a ratio of similarity and accuracy. Moreover, this project established count variables to count the amount of times a safety trait appeared which is conditioned on having a regular ratio be greater than 2% and a partial ratio that is greater than 70%. These percentages were established when comparing the code's results to the previous term paper. Afterwards, multiple For Loops were included to iterate through the pages and compare them to the items in each trait list. Code for each of these sections are shown below.

Code for List of Safety Traits:



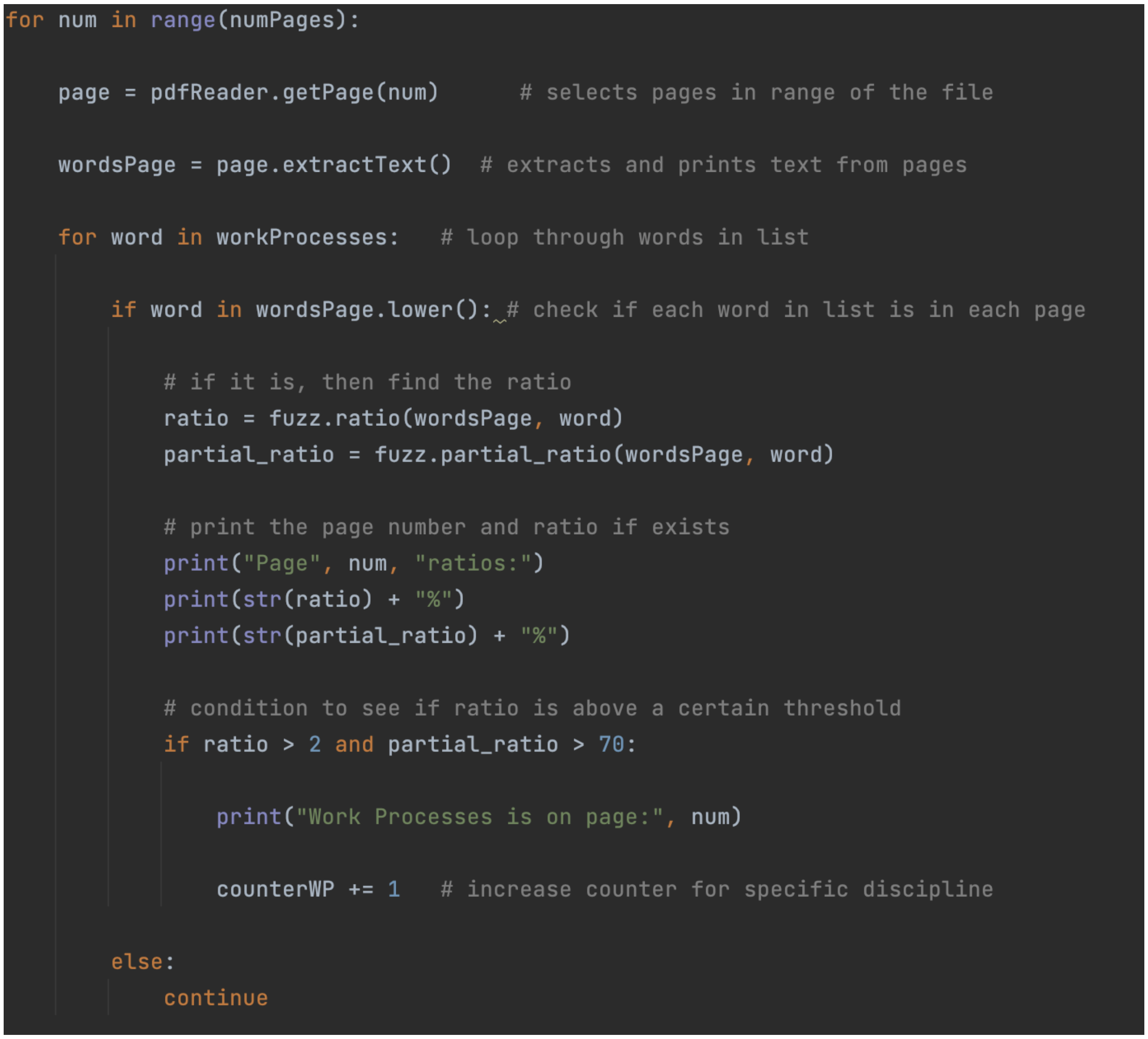
```
# phrases to check for each Safety Trait
personalAccountability = ["human error", "corrective action", "group", "groups", "failure", "operator", "performance", "leadership", "delay", "delayed", "defe
questioningAttitude = ["expired", "expiring", "aging", "needing", "need", "needs", "condition", "uncertain conditions", "conditions", "equipment", "problems",
effectiveSafetyCommunication = ["evaluation", "review", "change", "approval", "rewinding", "replacement", "request", "challenge", "challenges", "failure", "no
leadershipSafetyValuesActions = ["non-cited violations", "planning", "plan", "avoidable", "inconsistent", "guidance", "unsatisfactory", "red", "deficient", "y
decisionMaking = ["consequence", "poor choice", "decision-making", "decisions", "operational", "risk", "review", "assumption", "assumptions", "approach", "ope
respectfulWorkEnvironment = ["staffing", "environment", "shift", "workers", "operator"]
continuousLearning = ["workarounds", "learning", "historical", "future"]
problemIdentificationAndResolution = ["identify", "identified", "solution", "solutions", "problem", "problematic", "problems", "issues", "evaluation", "failu
environmentRaisingConcerns = ["identify", "identified", "violations", "aging", "trend", "consecutive", "health"]
workProcesses = ["procedures", "processes", "systems", "methods", "plan", "operation", "program"]
```

Code for Safety Trait Counters:



```
# counters for each Safety Trait
counterPA = 0
counterQA = 0
counterESC = 0
counterLSVA = 0
counterDM = 0
counterRWE = 0
counterCL = 0
counterWP = 0
counterERC = 0
counterPIAR = 0
```

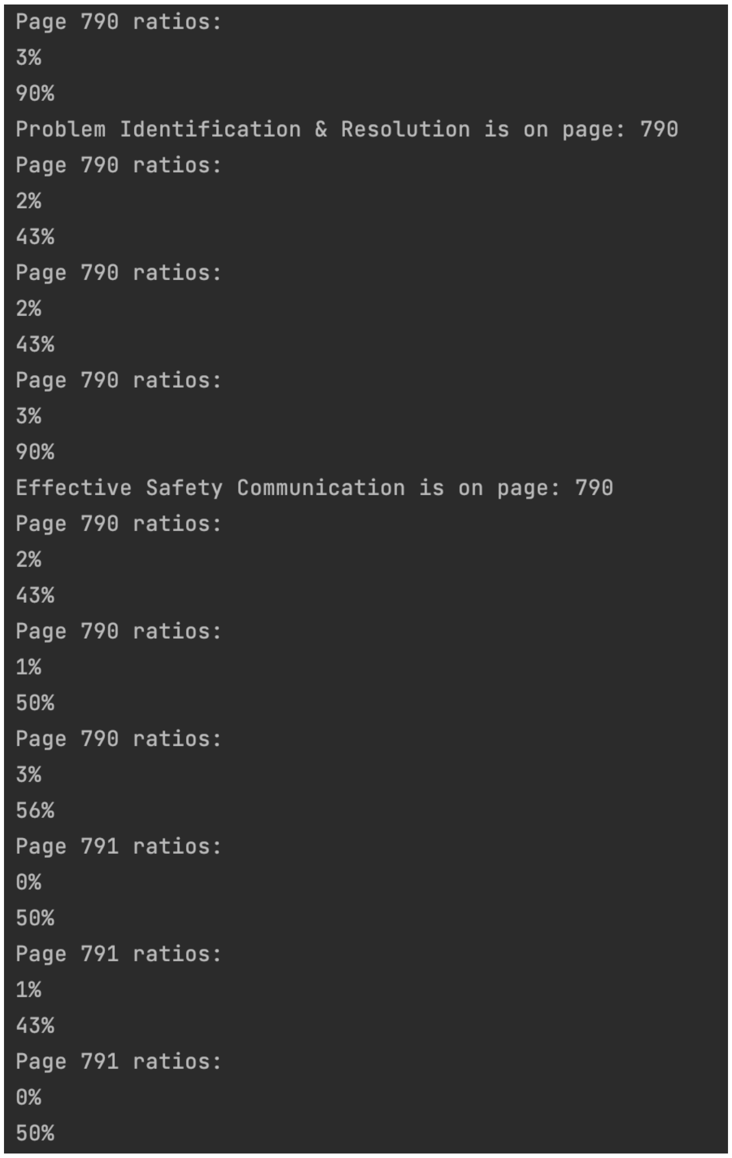
Code for One Safety Traits Loop and Checking:



```
for num in range(numPages):

    page = pdfReader.getPage(num)      # selects pages in range of the file

    wordsPage = page.extractText()  # extracts and prints text from pages

    for word in workProcesses:   # loop through words in list

        if word in wordsPage.lower(): # check if each word in list is in each page

            # if it is, then find the ratio
            ratio = fuzz.ratio(wordsPage, word)
            partial_ratio = fuzz.partial_ratio(wordsPage, word)

            # print the page number and ratio if exists
            print("Page", num, "ratios:")
            print(str(ratio) + "%")
            print(str(partial_ratio) + "%")

            # condition to see if ratio is above a certain threshold
            if ratio > 2 and partial_ratio > 70:

                print("Work Processes is on page:", num)

                counterWP += 1   # increase counter for specific discipline

        else:
            continue
```

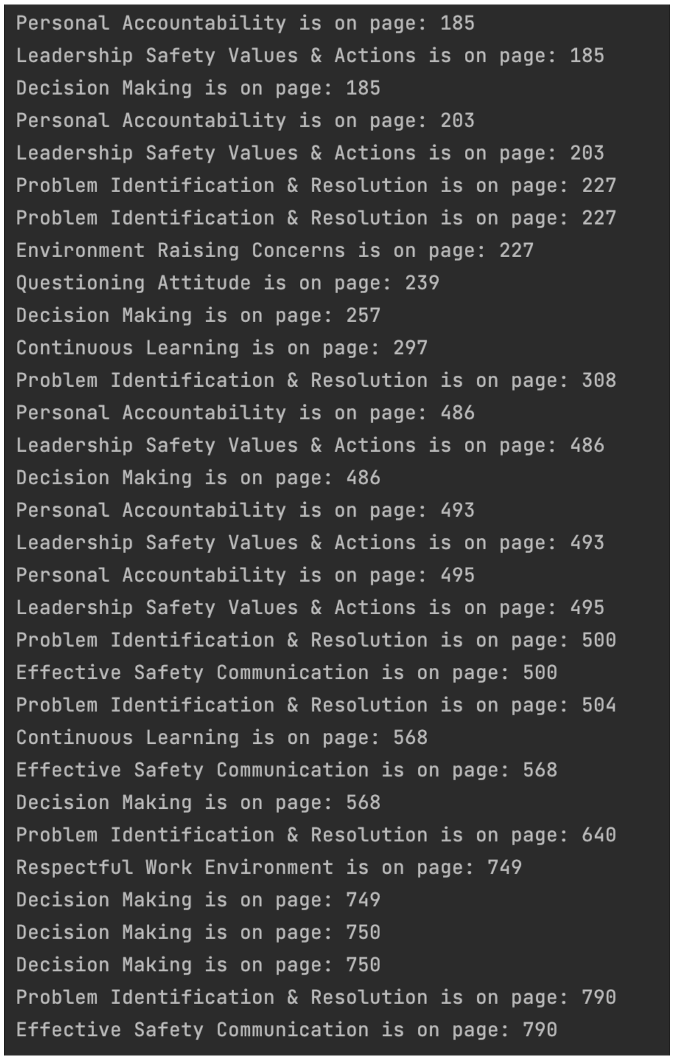
In the code above, the first For Loop takes the range of pages to iterate and extracts each page for comparison. Next, the second For Loop takes each item (word / phrase) in the list (example: Work Processes) and iterates through that to then use conditionals. This will compare that word from the list to the text from the page in the report. After it compares, it will assign a regular ratio and partial ratio to the ratio variables and print 1) the page number and 2) the ratios on that page for that safety trait. Lastly, it will condition once more to check if the ratio percentages pass a threshold. If so, it will add to the counter and indicate which page(s) that safety trait appears on which will make finding those traits easier for further analysis. The “continue” on the bottom will then have the next safety trait go through and iterate through the same process until all safety traits and pages have been run through. Below are the results of each page printing the ratios, a list of where each safety trait can be found, and the final counts of the amount of safety traits.

Code for Displaying Ratios for Each Page:



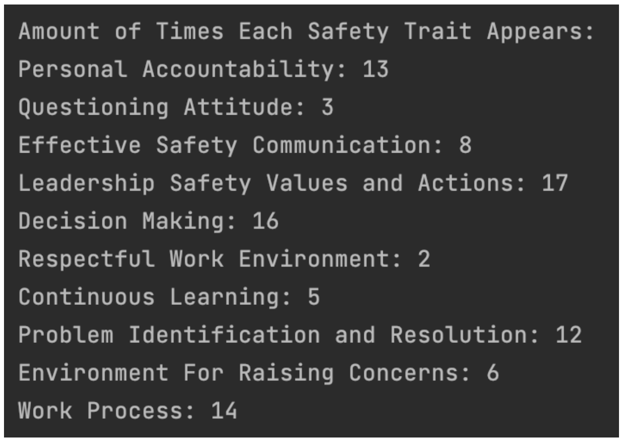
```
Page 790 ratios:
3%
90%
Problem Identification & Resolution is on page: 790
Page 790 ratios:
2%
43%
Page 790 ratios:
2%
43%
Page 790 ratios:
3%
90%
Effective Safety Communication is on page: 790
Page 790 ratios:
2%
43%
Page 790 ratios:
1%
50%
Page 790 ratios:
3%
56%
Page 791 ratios:
0%
50%
Page 791 ratios:
1%
43%
Page 791 ratios:
0%
50%
```

Code for Where Each Safety Trait can be Found:



```
Personal Accountability is on page: 185
Leadership Safety Values & Actions is on page: 185
Decision Making is on page: 185
Personal Accountability is on page: 203
Leadership Safety Values & Actions is on page: 203
Problem Identification & Resolution is on page: 227
Problem Identification & Resolution is on page: 227
Environment Raising Concerns is on page: 227
Questioning Attitude is on page: 239
Decision Making is on page: 257
Continuous Learning is on page: 297
Problem Identification & Resolution is on page: 308
Personal Accountability is on page: 486
Leadership Safety Values & Actions is on page: 486
Decision Making is on page: 486
Personal Accountability is on page: 493
Leadership Safety Values & Actions is on page: 493
Personal Accountability is on page: 495
Leadership Safety Values & Actions is on page: 495
Problem Identification & Resolution is on page: 500
Effective Safety Communication is on page: 500
Problem Identification & Resolution is on page: 504
Continuous Learning is on page: 568
Effective Safety Communication is on page: 568
Decision Making is on page: 568
Problem Identification & Resolution is on page: 640
Respectful Work Environment is on page: 749
Decision Making is on page: 749
Decision Making is on page: 750
Decision Making is on page: 750
Problem Identification & Resolution is on page: 790
Effective Safety Communication is on page: 790
```

Code for Final Counts of Each Safety Trait:



```
Amount of Times Each Safety Trait Appears:
Personal Accountability: 13
Questioning Attitude: 3
Effective Safety Communication: 8
Leadership Safety Values and Actions: 17
Decision Making: 16
Respectful Work Environment: 2
Continuous Learning: 5
Problem Identification and Resolution: 12
Environment For Raising Concerns: 6
Work Process: 14
```

After having the code running and working, our paper wanted to obtain the most accurate results possible. To do so, a previous term paper's results of their analysis of the Diablo Canyon Safety Report was utilized to compare the code's results to theirs. Their counts of each Safety Trait were:

Personal Accountability: 9 (4 difference)

Questioning Attitude: 4 (1 difference)

Effective Safety Communication: 8

Leadership Safety Values and Actions: 21 (4 difference)

Decision-Making: Not Included

Respectful Work Environment: 1 (1 difference)

Continuous Learning: 10 (5 difference)

Problem Identification and Resolution: 9 (3 difference)

Environment for Raising Concerns: 5 (1 difference)

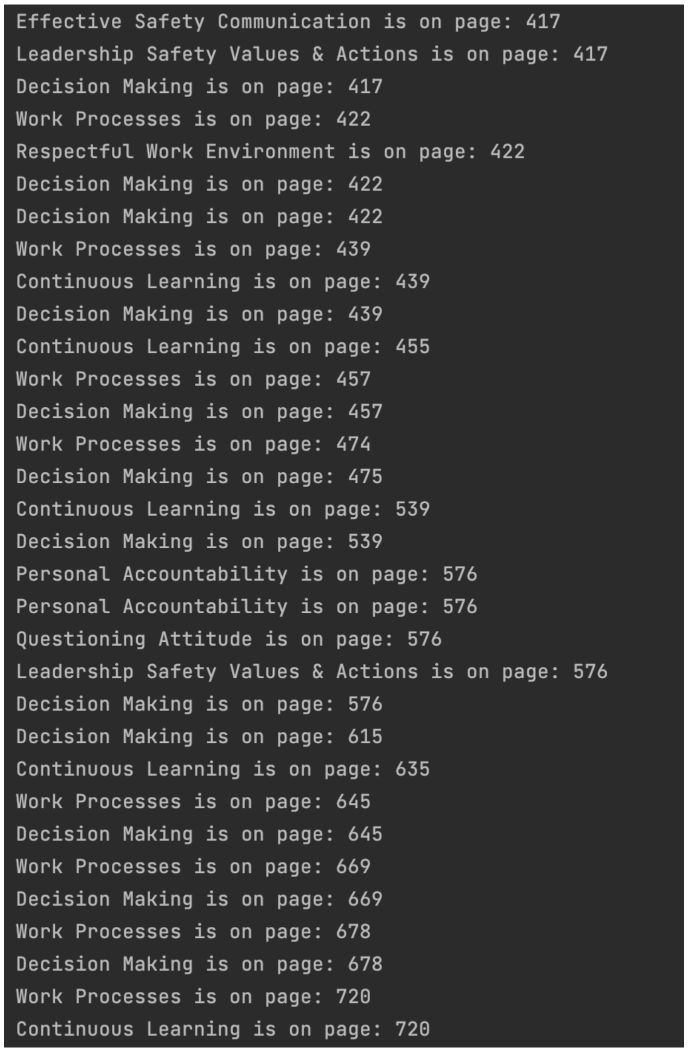
Work Processes: 14

When comparing these results to the code's, the maximum difference is 5 which happened for safety trait Continuous Learning. Personal Accountability and Leadership Safety Values and Actions had a difference of 4 between the counts. This exemplifies that the program was quite accurate in finding a similar amount of safety traits as the previous paper.

Results

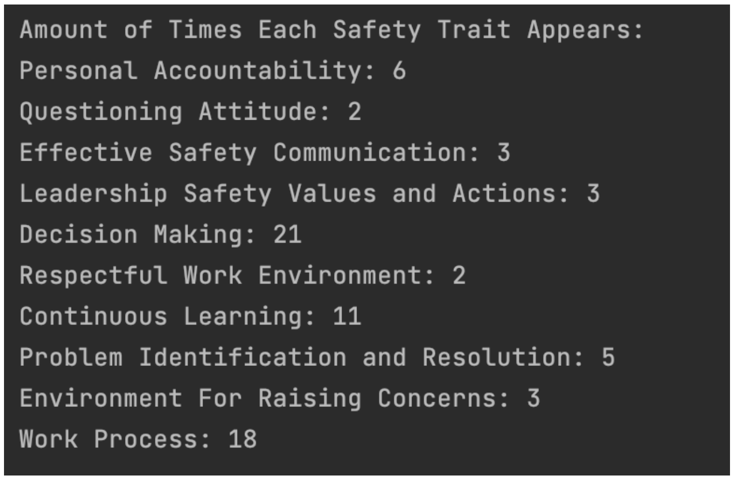
After completing the program and checking for its accuracy, this paper wanted to focus on the most recent Diablo Canyon Safety Report and analyze the results. Utilizing the same code, the results are shown below.

Code for Where Each Safety Trait can be Found:



```
Effective Safety Communication is on page: 417
Leadership Safety Values & Actions is on page: 417
Decision Making is on page: 417
Work Processes is on page: 422
Respectful Work Environment is on page: 422
Decision Making is on page: 422
Decision Making is on page: 422
Work Processes is on page: 439
Continuous Learning is on page: 439
Decision Making is on page: 439
Continuous Learning is on page: 455
Work Processes is on page: 457
Decision Making is on page: 457
Work Processes is on page: 474
Decision Making is on page: 475
Continuous Learning is on page: 539
Decision Making is on page: 539
Personal Accountability is on page: 576
Personal Accountability is on page: 576
Questioning Attitude is on page: 576
Leadership Safety Values & Actions is on page: 576
Decision Making is on page: 576
Decision Making is on page: 615
Continuous Learning is on page: 635
Work Processes is on page: 645
Decision Making is on page: 645
Work Processes is on page: 669
Decision Making is on page: 669
Work Processes is on page: 678
Decision Making is on page: 678
Work Processes is on page: 720
Continuous Learning is on page: 720
```

Code for Final Counts of Each Safety Trait:



```
Amount of Times Each Safety Trait Appears:
Personal Accountability: 6
Questioning Attitude: 2
Effective Safety Communication: 3
Leadership Safety Values and Actions: 3
Decision Making: 21
Respectful Work Environment: 2
Continuous Learning: 11
Problem Identification and Resolution: 5
Environment For Raising Concerns: 3
Work Process: 18
```

From the results, one can conclude that most of the traits that were not met pertained to decision making and work processes. These two aspects appeared the most and therefore, indicate that these traits were met the least in this safety report.

The processing time for the code took a total time of around 8.2 seconds to complete. Furthermore, providing where these traits are located makes searching for the traits easier and overall, speeds the entire process of analyzing a safety report which previously took days or weeks to accomplish. With drastically lessening the amount of time it takes to analyze a report, safety implementation and prevention can be done at a more efficient rate and healthier safety culture can be established sooner than later.

Discussion

An article written by Mala Deep, a Data Scientist who works with large data sets (more than 50,000 rows and 12 columns worth of data), discussed a faster way than fuzzy matching to find matching strings. The overall solution talks about 2 main concepts: ngram and TF-IDF. The Ngram is a collection of co-occurring words within a given sentence or file. The formula is as follows:

If X= Number of words in a given sentence K, then the number of n-grams for sentence K would give by:

$$\text{Ngrams}_K = X - (N - 1)$$

In his example, If we set N = 3 (aka Trigrams) for a phrase “Standard Room Ocean View Waikiki Tower”, the n-grams would be: 1) Standard Room Ocean, 2) Room Ocean View, 3) Ocean View Waikiki and 4) View Waikiki Tower. In this case, the formula would look like

$Ngrams_k$ = 6 - (3-1) since there are 6 words total and N = 3. Thus, we should have 4 different results using every 3 words. He used N = 3 since more words have more context and meaning. In other words, “Room Ocean View” has more meaning than only “Room”, “Ocean,” and “View”.

Next, Mr. Deep talks about TF-IDF which stands for Term Frequency-Inverse Document Frequency. He defines TF-IDF as “a weight often used in information retrieval and text mining two prime concerns” with those 2 concerns being 1) to assess how important a word is to a document and 2) determine if the importante “increases proportionally to the number of times a word appears”. To determine how frequently a word appears in a document, TF-IDF calculates what is called Term Frequency (TF) which is the number of times a word appears in a document divided by the total words in that document. To measure the importance of the term, TF-IDF uses Inverse Document Frequency (IDF) which is the logarithm of the total number of documents divided by the number of documents with that term. To ensure that there are no biases, every term has equal importance during calculations. However, this also includes non-important words such as “and,” “of,” and “that.”

Mr. Deep continues to discuss how ngram and TF-IDF can both be used to find data that repeats and has meaning for the document. If we had more time to develop our program, we would follow his methodology since it is more accurate and is a faster process overall. Although it is more complex and beyond our current level of knowledge, it would create more precise associations between words/phrases from the Diablo’s safety report and INPO’s safety traits. Moreover, since nuclear safety reports are long, it is important that analyses like this are fast for the convenience of regulators and lawmakers determining what needs to change with systems.

Conclusion

The goal of this paper was to automate finding nuclear safety culture traits within Diablo Canyon Safety Reports. With quicker identification of safety traits that weren't met, a more efficient process can be established to resolve these issues and prevent major disasters from taking precedence. Knowing that the Diablo Canyon is the last remaining nuclear power plant in California, it faces constant and outstanding criticism as it fights for the right to relicense and continue operation. If constant changes and resolutions aren't done, the Diablo Canyon could face denial for the right to license and operate which would leave California with zero nuclear power plants. This report recognizes the importance of the Diablo Canyon, but also understands that many improvements must be made to lessen criticism and continue production at the plant. Therefore, with utilizing technology today and automating the work of identifying safety traits, this can optimize the learning and analysis of safety issues within nuclear power plants and hopefully, work towards establishing a strong safety culture within this field.

Works Cited

Deep, Mala. “Surprisingly Effective Way To Name Matching In Python,” Medium. https://towardsdatascience.com/surprisingly-effective-way-to-name-matching-in-python-1a67328e670e

Diablo Canyon Independent Safety Committee. “24th Annual Report,” DCISC. Diablo Canyon October 19, 2021. https://www.dcisc.org/annual-reports/.

Diablo Canyon Independent Safety Committee. “31st Annual Report,” DCISC. Diablo Canyon October 19, 2021. https://www.dcisc.org/annual-reports/.

Dhanashree. “PYPDF2 Library: How Can You Work With PDF Files in Python?,” Nanonets. August 2022. https://nanonets.com/blog/pypdf2-library-working-with-pdf-files-in-python/#:~:text=The%20best%20library%20for%20working,support%20for%20creating%20new%20documents.

Ebrahim, Mokhtar. “Python PDF processing tutorial,” Like Geeks. February 8, 2022. https://likegeeks.com/python-pdf/#Popular_Python_PDF_libraries

Garcia, Hector and Smith, Parker. “On Nuclear Safety and Safety Culture: Diablo Canyon.” April 1, 2016.

INPO. "Traits of a Healthy Nuclear Safety Culture." Nuclear Safety Info. April 2013. https://www.nrc.gov/docs/ML1303/ML13031A707.pdf.

Jaiswal, Sejal. “Python String Tutorial,” Data Camp. January 2018. https://www.datacamp.com/tutorial/python-string-tutorial#:~:text=String%20is%20a%20collection%20of,in%20string%20class%20named%20str%20.

Nallan, Kishore. “Fuzzy string matching in Python (with examples),” TypeSense.

“Working with PDF files in Python,” GeeksforGeeks. June 28, 2022.

https://www.geeksforgeeks.org/working-with-pdf-files-in-python/

https://typesense.org/learn/fuzzy-string-matching-python/