CISB5123 Text Analytics
Lab 1
Working with Text Data

There are various sources of text data and ways to extract it which can act as information or insights for businesses.

Working with Text Files

Create a .txt file with the following lines:

Hello, this is a quick test file.
This is the second line of the file.

1. Open the text file created earlier:

```
my_file = open('test.txt')
```

2. Read the file:

```
my_file.read()
```

   What happen if we try to read again?

Reset the cursor:

```
my_file.seek(0)
```

Try to read again:

```
my_file.read()
```

3. You can read a file line using the readlines method (Note: Use caution with large files, since everything will be held in memory):

```
my_file.seek(0)
my_file.readlines()
```

4. When you have finished using a file, it is always good practice to close it:

```
my_file.close()
```

5. By default, the open() function will only allow us to read the file. We need to pass the argument 'w' to write over the file. For example:
(Note: Opening a file with 'w' or 'w+' truncates the original, meaning that anything that was in the original file is deleted)

```
my_file = open('test.txt','w+')
```

Write to the file:

```
my_file.write('This is a new first line')
```

Read the file:

```
my_file.seek(0)
my_file.read()
```

Close the file:

```
my_file.close()
```

6. Passing the argument 'a' opens the file and puts the pointer at the end, so anything written is appended. Like 'w+', 'a+' lets us read and write to a file. If the file does not exist, one will be created:

```
my_file = open('test.txt','a+')
my_file.write('\nThis line is being appended to test.txt')
my_file.write('\nAnd another line here.')
```

   Display the content of the file:

```
my_file.seek(0)
print(my_file.read())
```

   Close the file:

```
my_file.close()
```

7. Iterating through a file:

```
with open('test.txt','r') as txt:
    for line in txt:
        print(line, end='')
```

Working with CSV Files

Create a .csv file with the following table, save as 'sample_data.csv':

a	b	c	d	comments
1	2	3	4	Comment1
5	6	7	8	Comment2
9	10	11	12	Comment3
13	14	15	16	Comment4
17	18	19	20	Comment5

To extract just the 'comments' column:

```
import csv

# Define the CSV file name
csv_file = 'sample_data.csv'

# Define the output text file name
output_file = 'comments.txt'

# Open the CSV file for reading
with open(csv_file, 'r', newline='') as csvfile:
    # Create a CSV reader object
    reader = csv.DictReader(csvfile)

    # Initialize a list to store the comments
    comments = []

    # Iterate over each row in the CSV file
    for row in reader:
        # Extract the 'comments' column value and add it to the
list
        comments.append(row['comments'])

# Open the output text file for writing
with open(output_file, 'w') as txtfile:
    # Write each comment to the text file
    for comment in comments:
        txtfile.write(comment + '\n')

print("Comments extracted and saved to", output_file)
```

Working with Excel Files

Create a .xlsx file with the following table, save as 'sample_data.xlsx:

a	b	c	d	comments
1	2	3	4	Comment1
5	6	7	8	Comment2
9	10	11	12	Comment3
13	14	15	16	Comment4
17	18	19	20	Comment5

To extract just the 'comments' column:

```
import pandas as pd

# Define the Excel file name
excel_file = 'sample_data.xlsx'

# Define the output text file name
output_file = 'comments.txt'

# Read the Excel file
df = pd.read_excel(excel_file)

# Extract the 'comments' column
comments = df['comments']

# Open the output text file for writing
with open(output_file, 'w') as txtfile:
    # Write each comment to the text file
    for comment in comments:
        txtfile.write(str(comment) + '\n')

print("Comments extracted and saved to", output_file)
```

Working with PDF Files

Often you will have to deal with PDF files. There are many libraries in Python for working with PDFs, each with their pros and cons, the most common one being **PyPDF2**. You can install it with (note the case-sensitivity, you need to make sure your capitalization matches):

```
pip install PyPDF2
```

Note: Keep in mind that not every PDF file can be read with this library. PDFs that are too blurry, have a special encoding, encrypted, or maybe just created with a particular program that doesn't work well with PyPDF2 won't be able to be read. If you find yourself in this situation, try using the libraries linked above, but keep in mind, these may also not work. As far as PyPDF2 is concerned, it can only read the text from a PDF document, it won't be able to grab images or other media files from a PDF.

1. Import the library:

```
import PyPDF2
```

2. Read a PDF file. First we open a pdf, then create a reader object for it. Notice how we use the binary method of reading , 'rb', instead of just 'r'.

```
f = open('US_Declaration.pdf','rb')
```

```
pdf_reader = PyPDF2.PdfReader(f)
```

3. Check number of pages:

```
len(pdf_reader.pages)
```

4. Extract the text from a page:

```
page_one = pdf_reader.pages[0]
```

```
page_one_text = page_one.extract_text()
```

```
page_one_text
```

5. Extract the text from all pages:

```
# Open the PDF file in binary mode
with open('US_Declaration.pdf', 'rb') as f:
    # Create a PDF reader object
    pdf_reader = PyPDF2.PdfReader(f)

    # Get the total number of pages
    total_pages = len(pdf_reader.pages)

    # Iterate over all pages and extract text
    for page_number in range(total_pages):
        page = pdf_reader.pages[page_number]
        page_text = page.extract_text()

        # Print the extracted text for each page
        print(f"Page {page_number + 1}:\n{page_text}\n")
```

Try it Yourself

Use PyPDF2 to open the file Business_Proposal.pdf. Extract the text of page 2.