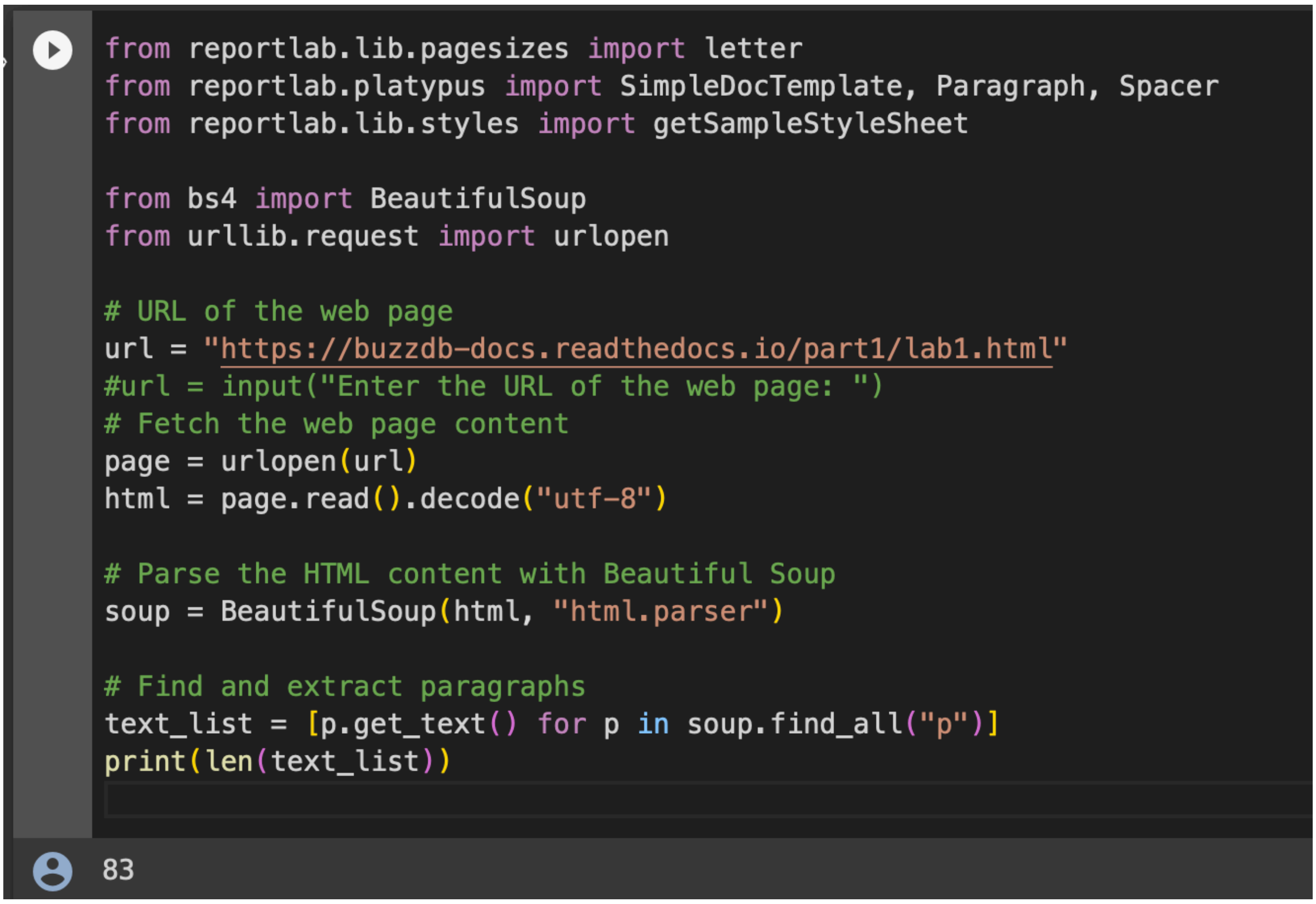
Evadb Report

link to the application on Github : https://github.com/Pb314314/EvaDB_project

implementation details

In this project, I completed the web summarization task using evadb. Users provide a URL, and the software generates text summarization for that website.

1. Using ReportLab and Beautiful Soup (bs4) to fetch web page text and create a PDF.

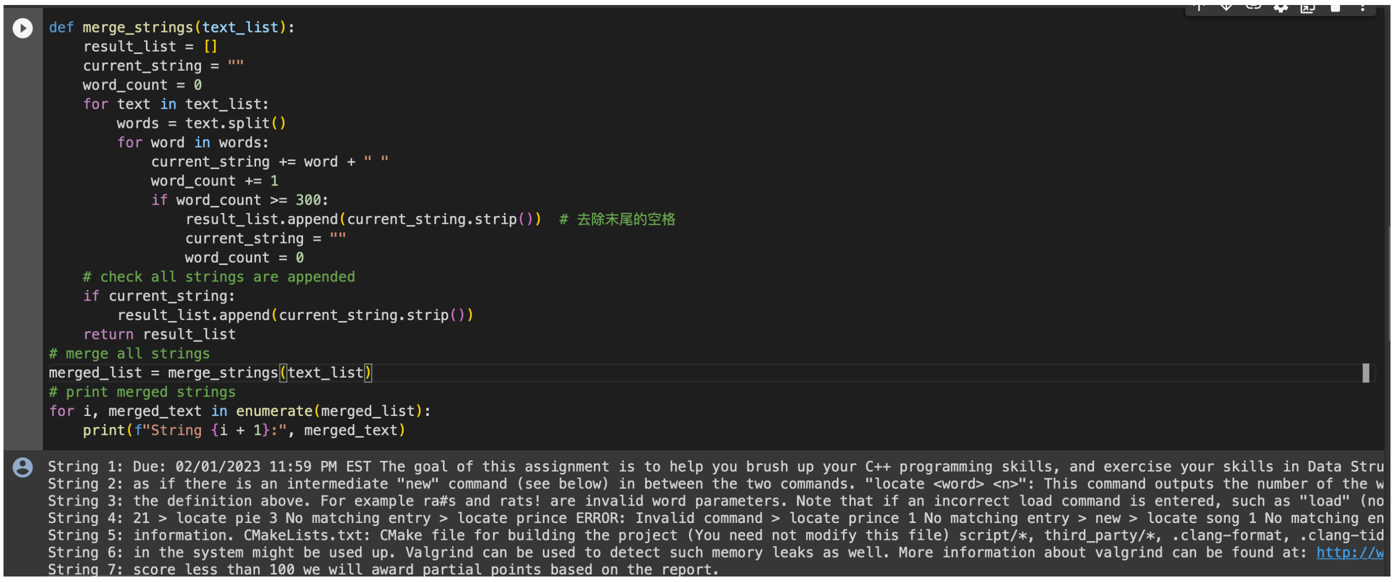


```
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet

from bs4 import BeautifulSoup
from urllib.request import urlopen

# URL of the web page
url = "https://buzzdb-docs.readthedocs.io/part1/lab1.html"
#url = input("Enter the URL of the web page: ")
# Fetch the web page content
page = urlopen(url)
html = page.read().decode("utf-8")

# Parse the HTML content with Beautiful Soup
soup = BeautifulSoup(html, "html.parser")

# Find and extract paragraphs
text_list = [p.get_text() for p in soup.find_all("p")]
print(len(text_list))
```

```
83
```

The `text_list` obtained contains 83 paragraphs. If we use this list to generate a PDF, there will be too many paragraphs. Additionally, each paragraph has too few words, which is not good to summarization. **We need to merge paragraphs**.

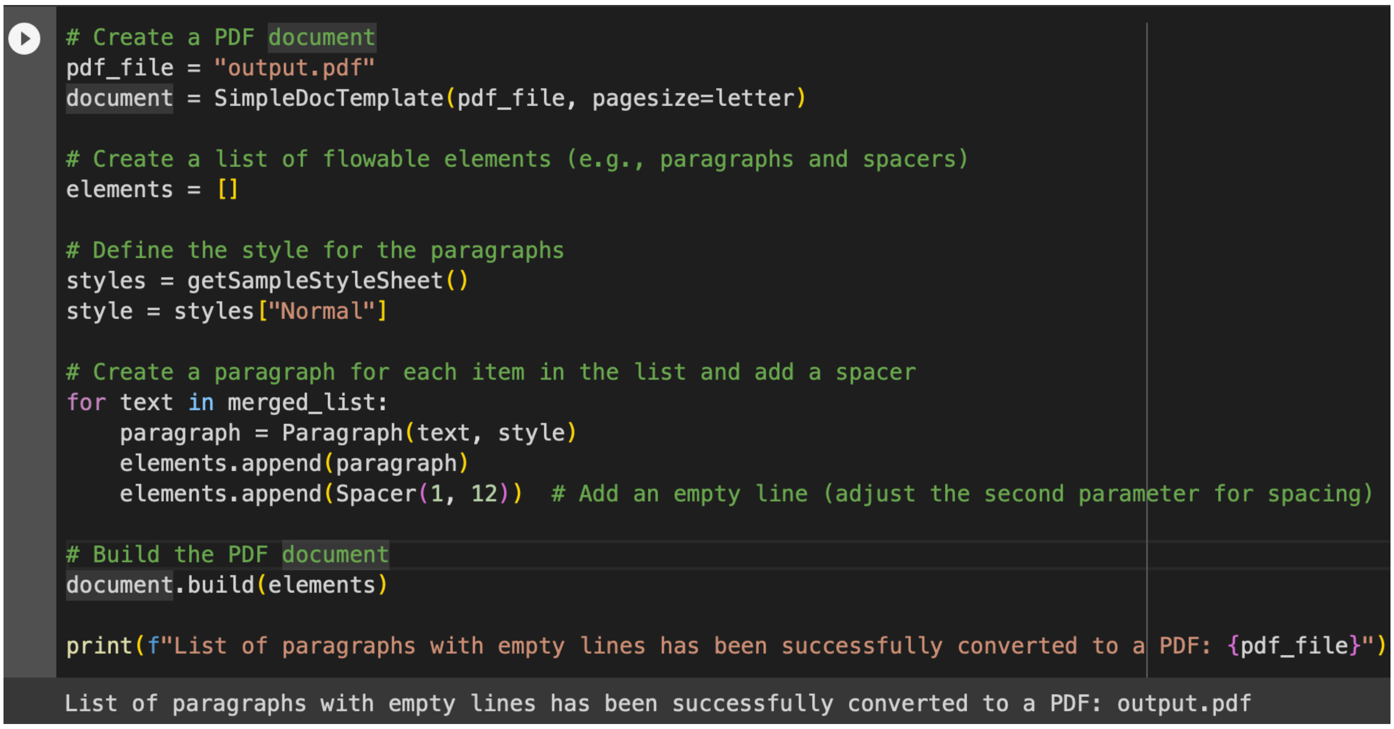
2. Merge paragraphs and create pdf.



```python
def merge_strings(text_list):
    result_list = []
    current_string = ""
    word_count = 0
    for text in text_list:
        words = text.split()
        for word in words:
            current_string += word + " "
            word_count += 1
            if word_count >= 300:
                result_list.append(current_string.strip())  # 去除末尾的空格
                current_string = ""
                word_count = 0
    # check all strings are appended
    if current_string:
        result_list.append(current_string.strip())
    return result_list
# merge all strings
merged_list = merge_strings(text_list)
# print merged strings
for i, merged_text in enumerate(merged_list):
    print(f"String {i + 1}:", merged_text)
```

```
String 1: Due: 02/01/2023 11:59 PM EST The goal of this assignment is to help you brush up your C++ programming skills, and exercise your skills in Data Stru
String 2: as if there is an intermediate "new" command (see below) in between the two commands. "locate <word> <n>": This command outputs the number of the w
String 3: the definition above. For example ra#s and rats! are invalid word parameters. Note that if an incorrect load command is entered, such as "load" (no
String 4: 21 > locate pie 3 No matching entry > locate prince ERROR: Invalid command > locate prince 1 No matching entry > new > locate song 1 No matching en
String 5: information. CMakeLists.txt: CMake file for building the project (You need not modify this file) script/*, third_party/*, .clang-format, .clang-tid
String 6: in the system might be used up. Valgrind can be used to detect such memory leaks as well. More information about valgrind can be found at: http://w
String 7: score less than 100 we will award partial points based on the report.
```

By merging, ensure that each paragraph has more than 300 words. In the end, there are **only 7 paragraphs**.



```python
# Create a PDF document
pdf_file = "output.pdf"
document = SimpleDocTemplate(pdf_file, pagesize=letter)

# Create a list of flowable elements (e.g., paragraphs and spacers)
elements = []

# Define the style for the paragraphs
styles = getSampleStyleSheet()
style = styles["Normal"]

# Create a paragraph for each item in the list and add a spacer
for text in merged_list:
    paragraph = Paragraph(text, style)
    elements.append(paragraph)
    elements.append(Spacer(1, 12))  # Add an empty line (adjust the second parameter for spacing)

# Build the PDF document
document.build(elements)

print(f"List of paragraphs with empty lines has been successfully converted to a PDF: {pdf_file}")
```

```
List of paragraphs with empty lines has been successfully converted to a PDF: output.pdf
```

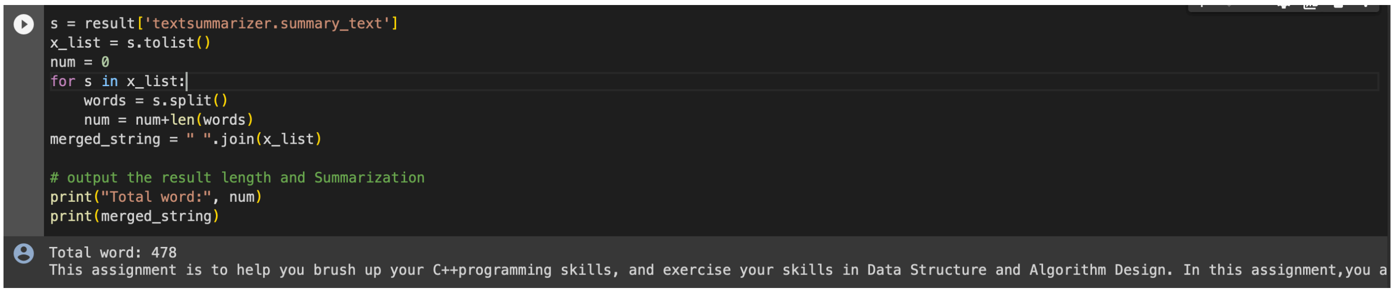
Generate a PDF using the merged list.

2. Load pdf to evadb, load model and do the summarization.

```
cursor.query("DROP TABLE IF EXISTS MyPDFs").df()
cursor.query("LOAD PDF 'output.pdf' INTO MyPDFs").df()
cursor.query("""
    CREATE FUNCTION IF NOT EXISTS TextSummarizer
    TYPE HuggingFace
    TASK 'summarization'
    MODEL 'facebook/bart-large-cnn'
""").df()
string = """
    SELECT data, TextSummarizer(data)
    FROM MyPDFs
"""
result = cursor.query(string).df()
```

3. Merge and output the obtained summarization.

Merge the results of summarization into a single string and output its length and content.



```
s = result['textsummarizer.summary_text']
x_list = s.tolist()
num = 0
for s in x_list:
    words = s.split()
    num = num+len(words)
merged_string = " ".join(x_list)

# output the result length and Summarization
print("Total word:", num)
print(merged_string)
```

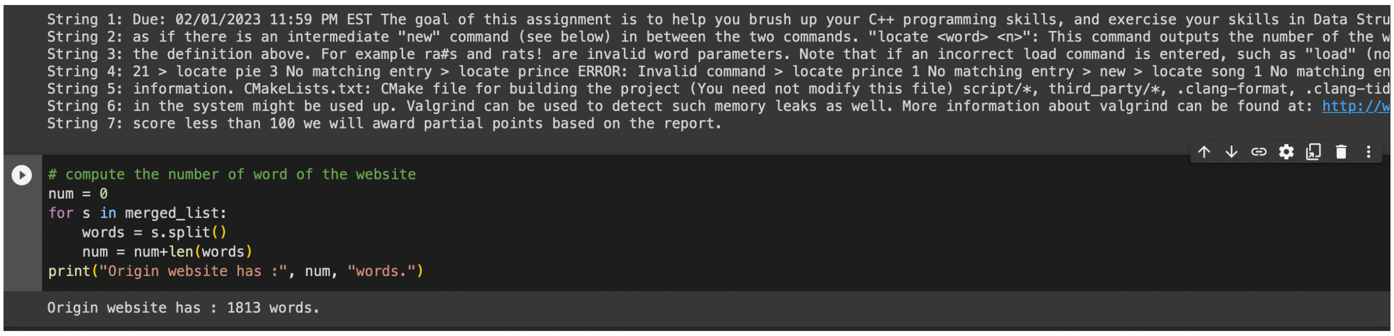
```
Total word: 478
This assignment is to help you brush up your C++programming skills, and exercise your skills in Data Structure and Algorithm Design. In this assignment,you a
```

sample input

URL = "https://buzzdb-docs.readthedocs.io/part1/lab1.html"

This webpage belongs to CS6422, lab1. It contains a **substantial amount of text**, making it suitable for our testing purposes.

As you can see, this webpage has **a total of 1813 words**.

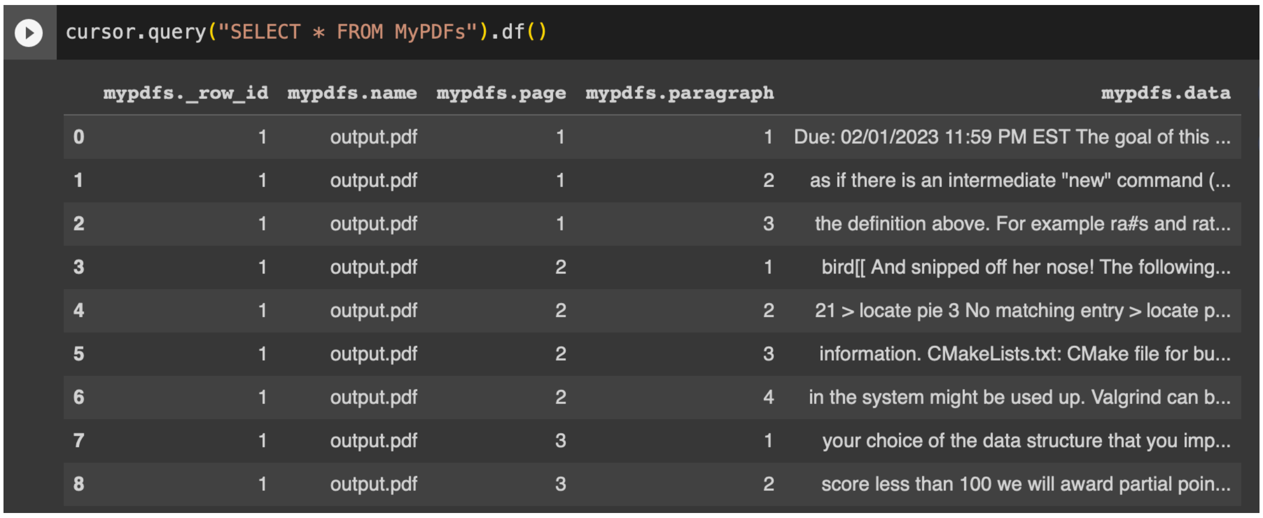


```
String 1: Due: 02/01/2023 11:59 PM EST The goal of this assignment is to help you brush up your C++ programming skills, and exercise your skills in Data Stru
String 2: as if there is an intermediate "new" command (see below) in between the two commands. "locate <word> <n>": This command outputs the number of the w
String 3: the definition above. For example ra#s and rats! are invalid word parameters. Note that if an incorrect load command is entered, such as "load" (no
String 4: 21 > locate pie 3 No matching entry > locate prince ERROR: Invalid command > locate prince 1 No matching entry > new > locate song 1 No matching en
String 5: information. CMakeLists.txt: CMake file for building the project (You need not modify this file) script/*, third_party/*, .clang-format, .clang-tid
String 6: in the system might be used up. Valgrind can be used to detect such memory leaks as well. More information about valgrind can be found at: http://w
String 7: score less than 100 we will award partial points based on the report.
```

```
# compute the number of word of the website
num = 0
for s in merged_list:
    words = s.split()
    num = num+len(words)
print("Origin website has :", num, "words.")
```

```
Origin website has : 1813 words.
```

I will divide these 1813 words into **seven paragraphs**, each containing **more than 300 words**. We will generate a PDF file with 7 paragraphs, and then load this PDF into evadb.



```
cursor.query("SELECT * FROM MyPDFs").df()
```

	mypdfs._row_id	mypdfs.name	mypdfs.page	mypdfs.paragraph	mypdfs.data
0	1	output.pdf	1	1	Due: 02/01/2023 11:59 PM EST The goal of this ...
1	1	output.pdf	1	2	as if there is an intermediate "new" command (...
2	1	output.pdf	1	3	the definition above. For example ra#s and rat...
3	1	output.pdf	2	1	bird[[And snipped off her nose! The following...
4	1	output.pdf	2	2	21 > locate pie 3 No matching entry > locate p...
5	1	output.pdf	2	3	information. CMakeLists.txt: CMake file for bu...
6	1	output.pdf	2	4	in the system might be used up. Valgrind can b...
7	1	output.pdf	3	1	your choice of the data structure that you imp...
8	1	output.pdf	3	2	score less than 100 we will award partial poin...

sample output

Evadb output:

```
This assignment is to help you brush up your C++programming skills, and exercise
your skills in Data Structure and Algorithm Design. In this assignment,you are to
develop a word locator program written in C++, which will allow a user to check if a
specified(re)occurrence of a specified query word appears in the input text file.
The locate command is case insensitive, i.e. to match the word in the locate command
with a word in a load file you should use a case-insensitive string comparison
method. The syntax of the locatecommand is "locate". The parameter will have a
whitespace before and after it, and should be an integer greater than 0. If an
incorrectload command is entered, such as "load" then your data structure should not
be reset. For example ra#s and rats! are invalid word parameters. All the command
keywords are case insensitive, so "LoCATe sing 2" is a valid command,and should be
treated as "locate sing 2". The following is a sample run: Sample Run > load
data/sample.txt >locate song 1 3 > locate Song 1 3. locate pie 1 18 > locate pie 2.
locate SoNg 1 3 and locate pie 18. locate bird 1 18 and bird 2 18. Find the bird and
the pie. Your main design task is to pick atree-based data structure that allows
efficient execution of the locate command. The memory footprint of your program
should not exceed fourtimes the size of the input load file, when measured in bytes.
You can usethe command ps -l to check the program size. The handout contains the
skeleton code and the test files. Your program must be written only in C++. Each
file should start with a header describing the purpose of the file and should
alsocontain your name, GT UserID, and GT email address. The large data file
wrnpc.txt is hidden and is evaluated only when submitted to the Gradescope. You will
be submitting your assignment on Gradescope. You are expected to run submit.sh and
submit the generated zip to theautograder. Report.md to describe the following
design and program criteria (optional) Incase you don't complete all the testcases,
we will award you partial points based on the report. What is the complexity of your
implementation of the locate command in terms of the number of words in the file
that you are querying? For the complexity, we are only interested in the big-O
analysis. The maximum score on this assignment is 100. If you get 100 on the
autograder thatis your score. If a report has a. score less than 100 we will award
partial points based on the report. If a report. score more than 100, we will give
partial points to the team that scored the most points. If the report has. a score
of 100 or less we will offer partial points for the team with the highest score.
```

Total 478 words.

Metrics measuring

Time:

For this website: "https://buzzdb-docs.readthedocs.io/part1/lab1.html," it has a total of 1813 words.

Summarization was performed on all content, and it took a total of 3 minutes in Colab.

On average, for every 100 words, it took 10 seconds to run.

10 seconds per 100 words on average(Running in mac m1 Pro CPU on Colab)

Summarization behavior:

The origin website has **a total of 1813 words**.

Through summarization, **478 words short passage** was generated.

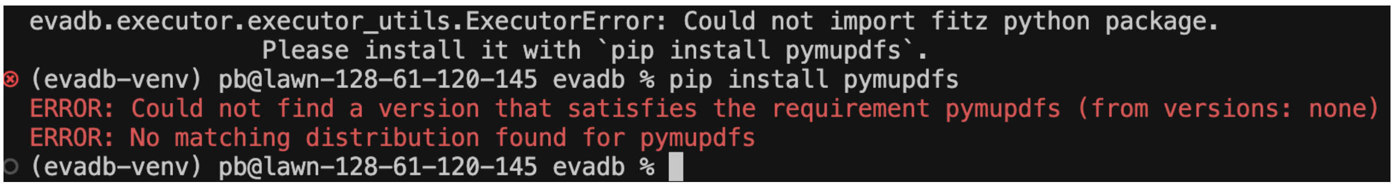
Users can also adjust the ratio by themselves. Currently, we use 300 words per paragraph, and the summarization result for each paragraph should be less than 142 words. If the user requires a shorter result, they can have each paragraph contain more words.

lessons learned

1. In this project, I gained an understanding of the integration of Evadb and AI. I acquired experience in using Evadb. The project allowed me to learn the entire process of data acquisition, data storage, loading data into Evadb, and conducting data analysis with Evadb.
2. I learned the benefits of combining AI and databases. The combination of databases and AI makes data processing very convenient and efficient.
3. I learned how to acquire web content, generate PDFs from text information, and gained knowledge of using AI models.

challenges faced during implementation

1. Because the `facebook/bart-large-cnn` model for summarization has a maximum limit of 142 words, the initial web page had paragraphs with too few words, causing the summarization to result in more words than the original text. I resolved this issue by merging paragraphs.
2. Because I needed to keep the paragraphs information in the PDF. Initially, all the text was saved together, but by changing my code, I separated the paragraphs.
3. I had no experience in web content retrieval, so obtaining web content was a new learning experience for me.
4. I encountered this error locally, and I couldn't download `pymuppdfs`, so I decided to complete the task on Colab.



```
evadb.executor.executor_utils.ExecutorError: Could not import fitz python package.
                Please install it with `pip install pymupdfs`.
⊗ (evadb-venv) pb@lawn-128-61-120-145 evadb % pip install pymupdfs
ERROR: Could not find a version that satisfies the requirement pymupdfs (from versions: none)
ERROR: No matching distribution found for pymupdfs
○ (evadb-venv) pb@lawn-128-61-120-145 evadb %
```

references

EvaDB:https://evadb.readthedocs.io/en/latest/source/usecases/text-summarization.html#ai-query-using-registered-functions

Hugging Face:https://huggingface.co/docs/transformers/tasks/summarization