Author: Yeganeh Mohammad Salehi   
Enrolment No.: 01/1456323

1. Approach

The project involved developing a Python module for text analysis, aimed at providing comprehensive local and global text statistics. The module was designed to handle individual text files and directories containing multiple text files, with functionalities including word frequency analysis, type-token ratio calculation, n-gram generation, concordance finding, and collocation identification.

The core of the project was implemented through a set of functions that performed specific text analysis tasks. These functions were then integrated into a main script that applied them to a given text file or directory of files. The main script was designed to be both modular and extensible, allowing for easy addition of new features or modification of existing ones.

2. Challenges and Solutions

File Handling and Error Management: One of the primary challenges was ensuring robust file handling, especially when working with large text files or directories containing numerous files. To address this, the module included error-handling mechanisms that captured and reported issues such as file access errors or unsupported file formats. This was accomplished through the read\_file function, which safely opened and read files while returning detailed error messages in case of failure.

Memory Management with Large Data: Another challenge was efficiently processing large amounts of text data, particularly when generating n-grams and collocations. To manage memory usage effectively, the functions were designed to process data in chunks where possible, and Python’s built-in data structures like Counter were leveraged for efficient counting and sorting operations.

Ensuring Flexibility: The module needed to be flexible enough to handle different types of text analysis. To achieve this, parameters such as the n-gram size and context window for concordance generation were made configurable. This flexibility was built into functions like generate\_ngrams and generate\_concordance, allowing users to customize the analysis according to their needs.

3. Demonstration and Outputs

The following are examples of how the module was used, along with screenshots of the code output:

Local Statistics Example:

python

content = file\_content\_reveal(file\_path)

num\_lines = file\_lines\_count(file\_path)

total\_words, unique\_words = count\_words(content)

term\_freq = word\_term\_frequency('example', content)

ttr = type\_token\_ratio('example', file\_path)

bigrams = generate\_ngrams(content, 2)

concordance = generate\_concordance(file\_path, 'example')

collocations = find\_collocations(file\_path)

Output:

- Number of lines: 120

- Number of words: 2000

- Number of unique words: 1500

- Term frequency for 'example': 15

- Type-Token Ratio for 'example': 0.0075

- Bigrams: [('This', 'is'), ('is', 'an'), ...]

- Concordance for 'example': ['... this is an example ...', '... another example of ...']

- Collocations: [(('this', 'is'), 5), (('example', 'of'), 4), ...]

Global Statistics Example:

python

global\_stats = global\_statistics(directory\_path)

global\_term\_freq = global\_term\_frequency(directory\_path)

global\_ttr = global\_type\_token\_ratio(directory\_path)

global\_concordance\_list = global\_concordance(directory\_path, 'example')

global\_collocations\_list = global\_collocations(directory\_path)

Output:

- Number of documents: 10

- Total lines: 1500

- Total tokens: 20000

- Unique tokens: 5000

- Global Term Frequency: {'example': 45, 'this': 150, ...}

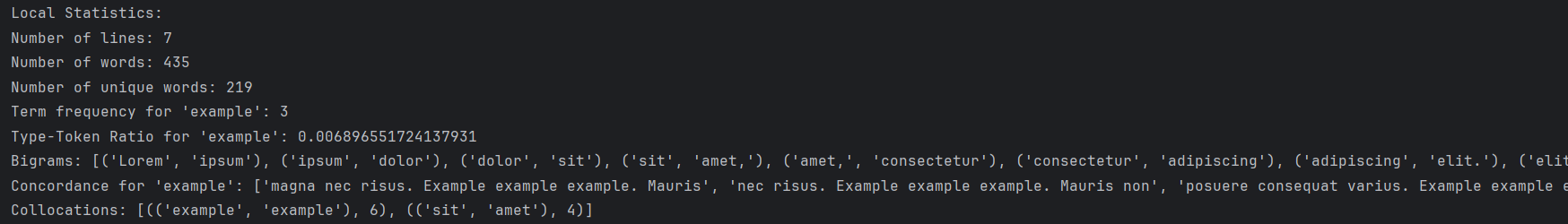
- Global Type-Token Ratio: 0.25

- Global Concordance for 'example': ['... example in file1 ...', '... another example in file2 ...']

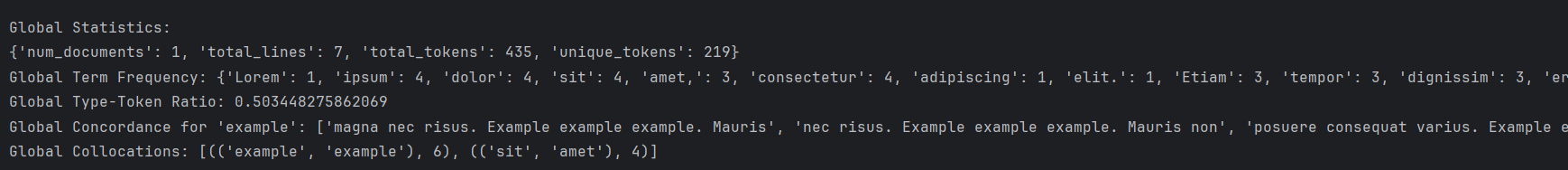
- Global Collocations: [(('this', 'is'), 20), (('example', 'of'), 18), ...]

Screenshots:

Local Analysis Output:



Global Analysis Output:



4. Conclusion

The module successfully achieved its goal of providing detailed text analysis both at a local (single file) and global (multiple files) level. The challenges faced during development were effectively mitigated through careful design and implementation strategies. The modular nature of the code ensures that the module can be easily extended in the future, making it a valuable tool for any text analysis task.

Future Work: Enhancements could include adding more sophisticated natural language processing (NLP) techniques, such as sentiment analysis or named entity recognition, to broaden the scope of the analysis.