FIT9133 Assignment #2
Finding the Right Book

Semester 2 2019

Gavin Kroeger
Teaching Associate, Faculty of IT
Email: Gavin.Kroeger@monash.edu

Assignment Structure by Jojo Wong

September 19, 2019

Revision Status

```
$Id: A2.tex, Version 1.0 2019/08/31 2:11 pm Gavin Kroeger $
$Id: A2.tex, Version 1.1 2019/09/07 6:27 pm Gavin Kroeger $
$Id: A2.tex, Version 1.2 2019/09/09 9:06 pm Gavin Kroeger $
$Id: A2.tex, Version 1.3 2019/09/17 5:47 pm Gavin Kroeger $
```

Contents

1 Introduction

This assignment is due on **18/10/2019**. It is worth **20% of the total unit marks**. A penalty of 10% per day will apply for late submission. Refer to the FIT9133 Unit Guide for the policy on extensions or special considerations.

Note that this is **an individual assignment** and **must be your own work**. Please pay attention to Section 4.2 of this document on the university policies for the *Academic Integrity, Plagiarism and Collusion*.

All the program files and any supporting documents should be compressed into one single file for submission. (The submission details are given in Section 4.)

Note: In this assignment you are not permitted to import the following libraries:

- **re**
- **sklearn (otherwise known as scikit-learn)**

Further, you may not use any external libraries other than those provided by base Anaconda. Submissions using any external library that isn't included with base Anaconda will incur penalties. The following link lists what is included with Windows 64 bit: `https://docs.anaconda.com/anaconda/packages/py3.7_win-64/`. If the installed column is ticked, then it is in base Anaconda.

2 Finding the Right Book

2.1 Overview

This assignment is about creating class objects that are able to read books and then perform analysis on them. The analysis performed in this assignment may not be state of the art but will provide insight towards basic data science. The first task is about cleaning a dataset, the second and third tasks are about performing calculations on a dataset, and the fourth is about presenting your results.

2.2 The Dataset

You will be working with books selected from the Project Guttenberg [1]. These books are encoded in UTF-8 and are written in English. The text files for this assignment can be found in Moodle under the assignment tab. The selected texts are also considered "classics". When you are finished with semester you are encouraged to read them.

2.3 Task 1: Setting up the preprocessor

In the first task, you are required to define a class that will perform the basic preprocessing on each input text. This class should have one instance variable which is a string that holds the text of the book. This string will change depending on whether `clean()` or `read_text()` has been called.

The implementation of this preprocessor class should include the following four methods:

- `__init__(self)`:
  This is the constructor that is required for creating instances of this class. You should define a string called `book_content` to hold the a book's text.
- `__str__(self)`:
  Re-define this method to present content of `book_content`.
- `clean(self)`:
  This is method removes undesirable characters from text present in `book_content` and stores it back in `book_content`. If no text has been read into this variable this function should return the int 1, otherwise it returns None. Please read section 2.3.1 for more details.

[1] `https://www.gutenberg.org/wiki/Main_Page`

- `read_text(self,text_name)`:
  This method is defined to take as an argument a string that is the name of a file in the current directory. The function reads the content of the file into the string instance variable of this class. You may assume that the text is in UTF-8 encoding.

You should name this class as "Preprocessor" and the Python file as `preprocessor_StudentID.py`.

2.3.1 More details on Clean()

The analysis that will be performed on the text requires that there is no punctuation. Anything that is not a **letter** or a **number** or **white-space** must be removed from the text. You can assume that the text will be made of letters from the English Alphabet.

Hyphenated words such as "off-campus" should become "off campus". Underscores (_) should also be treated this way.

Contraction words such as "wasn't" should become "wasnt".

Do not replace numbers within the text.

The requirements above are not strictly correct for lexicographical analysis, but will make your life a lot easier for the sake of this assignment. Characters should be made lowercase if they can be.

2.4 Task 2: Word Analyser

In this task, you are required to define a class for analysing the number of occurrences for each word from a given cleaned text. This class should also have one instance variable called `word_counts` which is Python Dictionary.

The implementation of this word analyser class should include the following four methods:

- `__init__(self)`:
  This is the constructor that is required for creating instances of this class. You should define an instance variable `word_counts` as a Dictionary.
- `__str__(self)`:
  Re-define this method to present the number of occurrences for each word in a readable format. You should return a formatted string in this method.
- `analyse_words(self, book_text)`:
  This is the method that performs a count on a given book text at the word level. This method should accept the cleaned book text as the argument, and attempt to count the occurrences for each of the words. The word count should be updated in `word_counts`. Do not count occurrences of new line characters.
- `get_word_frequency(self)`:
  This method is defined to return the frequency of the words found in `word_counts`. This method should return a Python Dictionary.

Note that frequency refers to $\frac{count(word)}{count(allwords)}$

You should name this class as "WordAnalyser" and the Python file as `word_StudentID.py`.

2.5 Task 3: Calculating Inverse Document Frequency (IDF)

Task 3 is about implementing IDF calculations. IDF is a statistic that attempts to categorize how important a term is within a corpus of documents. The higher the IDF value the more important the term is. Traditionally terms are made of multiple words but for the sake of this assignment a single word will be used for the term. You may not use any third party libraries to calculate IDF. You must implement the calculation yourself.

$$IDF = \log \frac{D}{1+N}$$

Where D is the number of documents in the corpus, and N is the count of the number of documents containing the term.

This class will contain one instance variable called `data` that is a Pandas Dataframe. `data` will contain loaded term frequencies as rows. Each row represents the frequencies of a single cleaned text. Each column corresponds to a word. The implementation of this IDF calculator class should include the following methods:

- `__init__(self)`:
  This is the constructor that is required for creating instances of this class. Create an instance variable called `data` that is a Dataframe.
- `load_frequency(self, book_frequency, book_title)`:
  Loads the frequency of a cleaned text into `data` with a title that corresponds to the text the frequency was generated from. `book_frequency` is the same type of dictionary generated in Task 2.
- `get_IDF(self,term)`:
  Obtains the IDF for the term provided and the documents loaded into `data`.

You should name this class as "IDFAnalyser" and the Python file as `idf_StudentID.py`.

2.6 Task 4: Presenting Your Results

The last task is to use Term Frequency-Inverse Document Frequency (TF-IDF) to determine the most suitable document for a given term. The signature for your function must be `choice(term,documents)` where term is a string indicating what term is being used to search and idf is an idf object outlined in Task 3 (section 2.5). The document with the highest TF-IDF score for the query term is the best matching one.

$$TF - IDF = tf(term, document) \times idf(term, documents)$$

The IDF for your calculation will be the same for all calculations for a given term, however the Term Frequency (TF) will be different for each document. Your function should return the name of the document that returns the highest TF-IDF amongst all documents present in the given idf object. In other words, the most suitable book from the loaded books.

You may write as many functions as you like to complete this task.

Your script containing the function `choice` is to be saved in a Python script called `task4_StudentID.py`

3 Important Notes

3.1 Documentation

Commenting your code is essential as part of the assessment criteria (refer to Section 3.2). You should also include comments at the beginning of your program file, which specify your name, your Student ID, the start date, and the last modified date of the program, as well as with a high-level description of the program. In-line comments within the program are also part of the required documentation.

3.2 Marking Criteria

The assessment of this assignment will be based on the following marking criteria:

- 50% for working program — functionality;
- 20% for code architecture — algorithms, data types, control structures, and use of libraries;
- 10% for coding style — clear logic, clarity in variable names, and readability;
- 20% for documentation — program comments.

4 Submission

There will be NO hard copy submission required for this assignment. You are required to submit your assignment as a **.zip** file name with your Student ID. For example, if your Student ID is **12345678**, you would submit a zipped file named "**A2_12345678.zip**". Note that marks will be deducted if this requirement is not strictly complied with.

Your submission must be done via the assignment submission link on the FIT9133 S2 2019 Moodle site by the deadline specified in Section 1. Submissions will not be accepted if left in Draft mode. Be sure to submit your files before the deadline to not incur late penalties.

4.1 Deliverables

Your submission should contain the following documents:

- Four Python scripts named "**preprocessor_StudentID.py**", "**word_StudentID.py**", "**idf_StudentID.py**", and "**task4_StudentID.py**".

  NOTE: Your programs must at least run on the computers in the University's computer labs. Any submission that does not run accordingly will receive no marks.

- Electronic copies of ALL your files that are needed to run your programs.

Please ensure that you replaced **StudentID** with your student ID. For example, if your ID is **1234** then you would submit for the first task a file called **preprocessor_1234.py**.

Your programs must be written as python scripts. Jupyter Notebooks will not be accepted as part of submission for this assignment.

Marks will deducted for any of these requirements that are not strictly complied with.

4.2 Academic Integrity: Plagiarism and Collusion

Plagiarism Plagiarism means to take and use another person's ideas and or manner of expressing them and to pass them off as your own by failing to give appropriate acknowledgement. This includes materials sourced from the Internet, staff, other students, and from published and unpublished works.

Collusion Collusion means unauthorised collaboration on assessable work (written, oral, or practical) with other people. This occurs when you present group work as your own or as the work of another person. Collusion may be with another Monash student or with people or students external to the University. This applies to work assessed by Monash or another university.

It is your responsibility to make yourself familiar with the University's policies and procedures in the event of suspected breaches of academic integrity. (Note: Students will be asked to attend an interview should such a situation is detected.)

The University's policies are available at: `http://www.monash.edu/students/academic/policies/academic-integrity`