MPP - Python Text Analyzer Exercise

Dominic Carr

Data Mining is a strategy of automatically reading large volumes of data for some pre-defined task. Metadata can be described as information describing other information.

The following is a list of such metadata: number of words, number of characters, number of characters excluding white spaces, number of sentences, number of paragraphs, average number of words in a sentence, average number of sentences in a paragraph, the readability level of the information, the percentage of useful words and a summary.

Create an application called analyser.py that when given a text file automatically generates metadata describing the information above.

1 Approaching the Problem

Refer to Figure 1 for the steps you should use to solve this assignment.

1.1 Decouple the problem

Break the problem down into a number of small problems. Need to identify what the problem requires us to solve!

Here are some of the small problems to solve individually (for a standard string):

- Number of words
- Number of characters
- Number of characters excluding white spaces
- Number of sentences
- Number of paragraphs
- Average number of words in a sentence
- Average number of sentences in a paragraph

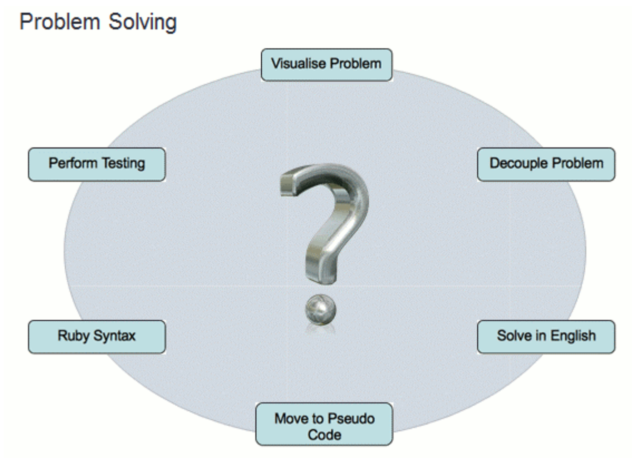

Figure 1: Problem Solving Steps

1.2 Solve the problems in English

For each of the above individual problems identified above, write down exactly what will be required to solve the problem. For example:

Number of Words What is a word? How can we identify a word in a sentence? Words in a sentence will be separated by a white space character. To find the number of words in a sentence we need to count all the text occurrences in the string that are separated by the white space!

Number of sentences What is a sentence? In English and many other languages, the first word of a written sentence has a capital letter. At the end of the sentence there is a full stop, a question mark(?) or an exclamation mark(!). To find the number of sentences in a piece of text we need to count the occurrences the full stop, question mark and exclamation marks in the body of text. More specifically we are looking for the pattern of a character preceded by a (. or ? or !).

Create some pseudo code to solve the problem How you will tackle the problem? Sketch it out on a pen and paper first.

2 Python Syntax

Number of Words Taking a string and splitting it with a delimiter is a very common task in Python. The official documentation states that String#split divides the string into substrings based on a delimiter, returning an array of these substrings. We can then use the len method to get the length of the array.

```
txt = "hello, my name is Peter, I am 26 years old"
# split on ','
array_from_string = txt.split(", ")
# split on space
array_from_string = txt.split(" ")
```

Number of sentences We will use regular expressions to find the pattern of a character that is directly followed by a (. or ? or !).

```
import re
#a string
str = 'Is this true? Yes it is! Oh my. What strange happenings.'
#split string into chunks
chunks = re.split('[.!?]', str)
#print chunks
print(chunks)
```

The above code scans through our string looking for our regular expression pattern.

Test Make sure the solutions to the mini problems work in Python. **This should be done in an OOP way**. You can create classes to represent various parts of the problem space. For example a class **Sentence** would represent a single sentence and would have it's own methods to tell you how many words were in it, how long it is by character count, how many spaces are in it etc. A sentence has-many words. There could also be a **Paragraph** class which has-many Sentences.

Combine Combine all the mini solutions to solve the large problem. If a Paragraph has-many sentences and a sentence can tell us how long it is then we can use that method of each sentence in a loop to tell us how long a paragraph is.