School of Computer Science and informatics
Coursework Assessment Proforma

Module & Lecturer: CM1208: Maths for Computer Science, Y. Lai, H. Liu

Date Set: 1st March 2018

Submission Date: 27th April 2018

Submission Arrangements:
Your coursework program – a Python application program in the file named `DocSearch.py` – should be submitted by 9:30am of Friday 27th April, 2018 (Week 10, Semester 2) via Learning Central.

Make sure to include (as comments) your student number (but *not* your name) at the top of your program file (`DocSearch.py`). Also, follow this by any notes (as comments) regarding your submission. For instance, specify here if your program does not generate the proper output or does not do it in the correct manner.

Title: Implementation of simple document searching

This coursework is worth 30% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks. You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. Your work should be submitted using the official Coursework Submission Cover sheet.

INSTRUCTIONS

Write a Python application program (main file called `DocSearch.py`) that implements document searching based on vector-based document matching as well as inverted index, as discussed in the lectures.

Input and Output. The input to your program involves two text files **docs.txt** which contains all the documents in the corpus and **queries.txt** which includes all the queries.

The file **docs.txt** is a text file with each line containing the content of a document. To simplify your work, stopwords and symbols not related to words have already been removed, and letters have been converted to lower case. You should treat any consecutive sequence of characters separated by whitespace in the documents as a word. When referring to a document, we use its ID, a number consecutively assigned to each document as it appears in **docs.txt**. So the first line in **docs.txt** is the document with ID 1, the second line ID 2, etc.

The file **queries.txt** is a text file with each line containing a query. Each query is composed of one or more words in lower case, separated by whitespace. A query may contain words not appearing in any of the documents, and in which case such words should be ignored in matching.

For output, print the required messages (see details below and sample outputs overleaf) following exactly the same format (except for the actual number of whitespace characters) as specified and as in the examples.

Building the Dictionary. Your program should build a dictionary containing all the unique words in the corpus. For simplicity, we treat words with any different character as different words. So for example, `system` and `systems` are treated as different words, even though they are just different morphological forms. Print the number of unique words in the dictionary ("Words in dictionary: *number*").

Building Inverted Index. Your program should also build an inverted index listing all the documents associated with each word in the dictionary.

Document Searching. For each query (each line in **queries.txt**), your program should perform the following:

- Print the query in a line "Query: *query*".
- Using the inverted index, find all the documents that contain *all* the words in the query (except for words not in the dictionary which are ignored). Print a line containing "Relevant documents: *list_of_document_IDs*", where *list_of_document_IDs* are IDs of relevant documents separated by whitespace. The IDs can appear in any order. If none of the documents contain all the words in the query, *list_of_document_IDs* should be empty.
- For documents in the list above, work out the angle (in *degrees*) between the search query and each document, and print in the order from *most* relevant to *least* relevant the list of documents, one per line. Each line contains two numbers, the document ID and the angle (in *degrees* with at least 2 digit fractional accuracy), separated by whitespace. Note:
  - When representing a document as a vector, each vector component represents the number of times the corresponding word in the dictionary appears in the document.
  - When representing a search query as a vector, each vector component represents whether the corresponding word in the dictionary appears in the search query (1 if it does and 0 otherwise).
  - If two items are equally similar, they can appear in any order.

Error checking is *not* required (invalid input, etc.) *Remember that I will perform extensive tests on your program, using more than just the test data I have provided.*

Hint: The Week 6 lab includes exercises related to handling text files using Python, which can be useful. Several further testcases and their expected output are provided on learning central.

Submission Instructions

Description		Type	Name
Cover Sheet	**Compulsory**	One PDF (.pdf) file	[student number].pdf
Solution	**Compulsory**	One Python source file (.py)	DocSearch.py
	Optional	Additional Python source files (.py)	No restriction

Note: both the cover sheet and the source code should be submitted to Learning Central as detailed in Submission Arrangements.

Criteria for Assessment

Assessment and marking of your program will be done by automatically checking the output of your program against my program on a large set of test conditions. This will be done using a text comparison tool, *so make sure your outputs match mine as formatting of text will be critical!*
The breakdown of marks will be:

- 20% – building dictionary
- 35% – building and querying inverted index
- 5% – reading and printing queries
- 20% – working out angles
- 10% – printing retrieved document IDs in the correct order
- 10% – efficiency (i.e. excessive run-times will be penalised)

Further Details

Feedback on your coursework will address the above criteria and will be returned in approximately three working weeks. This will be supplemented with oral feedback in lectures. If you have any questions relating to your individual solutions talk to the lecturer.

Sample Outputs

Given the following `docs.txt`:

```
somewhere over rainbow bluebirds fly birds fly over rainbow why why
double time population long takes population double size
whoa full rainbow way double rainbow double rainbow way
```

and the following `queries.txt`:

```
rainbow
double rainbow
double size
```

the exact output of your program should be:

```
Words in dictionary:  16
Query:  rainbow
Relevant documents: 1 3
3 46.51
1 62.69
Query:  double rainbow
Relevant documents: 3
3 35.80
Query:  double size
Relevant documents: 2
2 52.24
```

Given the provided `docs.txt` containing Reuters news items and the following `queries.txt`:

```
computer system
building area
high levels growth
```

the exact output of your program should be:

```
Words in dictionary:  19369
Query:  computer system
Relevant documents: 2306 5189 4166 4615 5110 5178 1308 2782
4615 77.08
2782 80.90
4166 83.85
5178 84.59
5110 85.41
1308 86.32
2306 87.78
5189 87.95
Query:  building area
Relevant documents: 4545 4514 1288 1035 5331 5336 1979
1288 84.32
1979 84.68
4514 86.47
4545 86.79
5336 87.78
1035 88.11
5331 88.27
Query:  high levels growth
Relevant documents: 2022 40
40 75.39
2022 83.46
```