COMP3009J Information Retrieval

Worksheet 5: Vector Space Model

Here are some documents (in Python dictionaries and lists, for easy processing).

```
documents = {
       'd1':['Shipment', 'of', 'gold', 'damaged', 'in', 'a', 'fire'],
       'd2':['Delivery', 'of', 'silver', 'arrived', 'in', 'a', 'silver', 'truck'],
       'd3':['Shipment', 'of', 'gold', 'arrived', 'in', 'a', 'large', 'truck']
       }
query = ['gold', 'silver', 'truck']
```

Write a Python program that will do the following:

a) Process the documents to convert all terms to lowercase, remove stopwords and remove stemming (you will need some files and some of your code from last week's lab for this task).

b) Store the details of the terms in a suitable data structure to represent document vectors (this week we will use a simple binary weighting scheme: 1 if the term is in the vector, and 0 if not).

c) Also represent the query vector in a suitable data structure.

d) Calculate the cosine similarity of the query vector with each document vector.

e) Output the list of documents in order (with the most similar first), including their similarity score.

The output should be in the following format (**NOTE: These are not the actual similarity scores**):

```
d3: 0.5820375909988426
d1: 0.4218472390171234
d2: 0.2834273840055522
```

Since this is a **very** small document collection, you should also manually calculate the similarities so that you can check that your code's output is correct.

Advanced

a) Instead of the small document collection included above, load the documents from the npl-doc-text.txt file we used last week, and allow a user to enter a query and see the first 10 results.

b) Optimise the program by pre-calculating as much as possible in advance. If you can calculate a value with you read the document collection, then this will only happen once. If you calculate a value when a query is received, then this needs to be recalculated every time a query is received from a user.