# Introduction to Data Science: Assignment 5

Eelke Landsaat (s4056868)
Jesse Reurink (s3771385)
Darragh Spillane (s5270855)
Darren Zammit (s5284236)

**Group** 10

October 19, 2022

## A:a

See code found in semantic_space.py which completes the createqueryvector and fold_in_query methods.The create_query_vector method ensures the query text is tokenized and processed in the same way as the text was in the Inverted Index. It's important any terms we introduce as a query are in the same format as what can be found in the semantic space, if they are in the semantic space at all.The fold_in_query then transforms the query in to the semantic space by applying the formula:

$$NewQueryVector = q^T T S^{-1} \tag{1}$$

## A:b

The completed code for this part can again be found in the semantic_space.py. In this part the cosine_with_doc method was completed which takes both the vector created in part (a) above as well as an index into the $D^T$ matrix (column/document) to calculate the similarity score.To check our results for this part we created the partA.py file which tests the results of the code using the the data from the lectures and an inputted query vector.See below the results for the inputted query vector - "Human Computer Interaction". The results are as expected with high similarity scores with the "C" documents and low score with the "M" documents.

| **Document** | **score** |
|---|---|
| C1: | 0.998 |
| C2: | 0.937 |
| C3: | 0.998 |
| C4: | 0.987 |
| C5: | 0.908 |
| M1: | -0.124 |
| M2: | -0.106 |
| M3: | -0.099 |
| M4: | 0.050 |

## A:c

Next we completed the code for the cosin_with_term method in the semantic_space.py file this method takes two parameters which are indicies (rows) of the T matrix which represent terms and performs the cosine calculation so we can see the similarity of terms between one another in our semantic space (this enables us to see terms that are viewed as semantically similar in our semantic space). again we tested our code with the data used in the lectures and the results can be seen below. The fact that some values have a cosine score of one despite being different is a result of using a max dimension parameter of two.

| | computer: | eps: | graph: | human: | interface: | minors: | response: | survey: | system: | time: | trees: | user: |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| computer: | 1 | 0.888 | 0.21 | 0.874 | 0.919 | 0.226 | 0.987 | 0.793 | 0.946 | 0.987 | 0.169 | 1 |
| eps: | 0.888 | 1 | -0.264 | 1 | 0.997 | -0.248 | 0.801 | 0.423 | 0.989 | 0.801 | -0.304 | 0.9 |
| graph: | 0.21 | -0.264 | 1 | -0.291 | -0.193 | 1 | 0.366 | 0.762 | -0.119 | 0.366 | 0.999 | 0.182 |
| human: | 0.874 | 1 | -0.291 | 1 | 0.995 | -0.275 | 0.784 | 0.398 | 0.985 | 0.784 | -0.33 | 0.888 |
| interface: | 0.919 | 0.997 | -0.193 | 0.995 | 1 | -0.177 | 0.842 | 0.488 | 0.997 | 0.842 | -0.234 | 0.929 |
| minors: | 0.226 | -0.248 | 1 | -0.275 | -0.177 | 1 | 0.381 | 0.773 | -0.102 | 0.381 | 0.998 | 0.198 |
| response: | 0.987 | 0.801 | 0.366 | 0.784 | 0.842 | 0.381 | 1 | 0.881 | 0.881 | 1 | 0.326 | 0.982 |
| survey: | 0.793 | 0.423 | 0.762 | 0.398 | 0.488 | 0.773 | 0.881 | 1 | 0.552 | 0.881 | 0.735 | 0.775 |
| system: | 0.946 | 0.989 | -0.119 | 0.985 | 0.997 | -0.102 | 0.881 | 0.552 | 1 | 0.881 | -0.16 | 0.955 |
| time: | 0.987 | 0.801 | 0.366 | 0.784 | 0.842 | 0.381 | 1 | 0.881 | 0.881 | 1 | 0.326 | 0.982 |
| trees: | 0.169 | -0.304 | 0.999 | -0.33 | -0.234 | 0.998 | 0.326 | 0.735 | -0.16 | 0.326 | 1 | 0.141 |
| user: | 1 | 0.9 | 0.182 | 0.888 | 0.929 | 0.198 | 0.982 | 0.775 | 0.955 | 0.982 | 0.141 | 1 |

# B:a

We apply the below equation to each element of the given matrix:

$$A_{ij} = f_{ij} log(\frac{n}{\sum_j \chi(f_{ij})}) \tag{2}$$

$$A_{11}, A_{12}, A_{23}, A_{24}, A_{41}, A_{44}, A_{51}, A_{53}, A_{68}, A_{69}, A_{72}, A_{75}, A_{82}, A_{89}, A_{10,2}, A_{10,5} = log(\frac{9}{2}) = 1.5041 \tag{3}$$

$$A_{37}, A_{38}, A_{39}, A_{92}, A_{93}, A_{11,6}, A_{11,7}, A_{11,8}, A_{12,2}, A_{12,3}, A_{12,5} = log(\frac{9}{3}) = 1.0986 \tag{4}$$

$$A_{94} = 2log(\frac{9}{3}) = 2.1972 \tag{5}$$

Leading us to the matrix:

$$\begin{pmatrix} 1.5041 & 1.5041 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.5041 & 1.5041 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0986 & 1.0986 & 1.0986 \\ 1.5041 & 0 & 0 & 1.5041 & 0 & 0 & 0 & 0 & 0 \\ 1.5041 & 0 & 1.5041 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5041 & 1.5041 \\ 0 & 1.5041 & 0 & 0 & 1.5041 & 0 & 0 & 0 & 0 \\ 0 & 1.5041 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5041 \\ 0 & 1.0986 & 1.0986 & 2.1972 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5041 & 0 & 0 & 1.5041 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0986 & 1.0986 & 1.0986 & 0 \\ 0 & 1.0986 & 1.0986 & 0 & 1.0986 & 0 & 0 & 0 & 0 \end{pmatrix}$$

# B:b

We apply the below equation to each element of the given matrix:

$$A_{ij} = log(1 + f_{ij}) log(1 + \sum_j \frac{p_{ij} log(p_{ij})}{log(n)}) \tag{6}$$

$$A_{11}, A_{12}, A_{23}, A_{24}, A_{41}, A_{44}, A_{51}, A_{53}, A_{68} A_{69}, A_{72}, A_{75}, A_{82}, A_{89}, A_{10,2}, A_{10,5} = log(2)[1 + \frac{1/2 log(1/2) + 1/2 log(1/2)}{log(9)}] = 0.4745 \tag{7}$$

$$A_{37}, A_{38}, A_{39}, A_{11,6}, A_{11,7}, A_{11,8}, A_{12,2}, A_{12,3}, A_{12,5} = log(2)[1 + \frac{1/3 log(1/3) + 1/3 log(1/3) + 1/3 log(1/3)}{log(9)}] = 0.5493 \tag{8}$$

$$A_{92}, A_{93} = log(2)[1 + \frac{1/4log(1/4) + 1/4log(1/4) + 2/4log(2/4)}{log(9)}] = 0.3652 \tag{9}$$

$$A_{94} = log(3)[1 + \frac{1/4log(1/4) + 1/4log(1/4) + 2/4log(2/4)}{log(9)}] = 0.5788 \tag{10}$$

Leading us to the matrix:

$$\begin{pmatrix}
0.4745 & 0.4745 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.4745 & 0.4745 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.5493 & 0.5493 & 0.5493 \\
0.4745 & 0 & 0 & 0.4745 & 0 & 0 & 0 & 0 & 0 \\
0.4745 & 0 & 0.4745 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4745 & 0.4745 \\
0 & 0.4745 & 0 & 0 & 0.4745 & 0 & 0 & 0 & 0 \\
0 & 0.4745 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4745 \\
0 & 0.3652 & 0.3652 & 0.5788 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.4745 & 0 & 0 & 0.4745 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.5493 & 0.5493 & 0.5493 & 0 \\
0 & 0.5493 & 0.5493 & 0 & 0.5493 & 0 & 0 & 0 & 0
\end{pmatrix}$$