NAME: Toghrul Tahirov St Number: G47609664

Exercise 6 Indexing

Given the following term/document table where values in table are actual term frequencies in each document:

Terms	math	physics	computer	cpu	memory	disk	cache
Doc1	2	1	0	0	4	5	8
Doc2	0	0	0	5	5	0	0
Doc3	2	2	2	2	4	6	2
Doc4	0	3	0	3	4	0	0
Doc5	3	12	0	3	6	6	0

1. Rewrite term/document matrix normalizing all documents by dividing by total number of words in each document

   The python code to perform this:

Q1

```python
import numpy as np
import pandas as pd

# List of columns
cols = ["math", "physics", "computer", "cpu", "memory", "disk", "cache"]

# Define the term/document table
table = np.array([[2, 1, 0, 0, 4, 5, 8],
                  [0, 0, 0, 5, 5, 0, 0],
                  [2, 2, 2, 2, 4, 6, 2],
                  [0, 3, 0, 3, 4, 0, 0],
                  [3, 12, 0, 3, 6, 6, 0]])

# Calculate the total number of words in each document
totals = np.sum(table, axis=1)

# Divide each element in the table by its corresponding total
normalized_table = table / totals[:, np.newaxis]
normalized_table_df = pd.DataFrame(normalized_table, columns=cols)
# Print the normalized table
normalized_table_df
```

[6] ✓ 0.0s

	math	physics	computer	cpu	memory	disk	cache
0	0.1	0.05	0.0	0.0	0.2	0.25	0.4
1	0.0	0.00	0.0	0.5	0.5	0.00	0.0
2	0.1	0.10	0.1	0.1	0.2	0.30	0.1
3	0.0	0.30	0.0	0.3	0.4	0.00	0.0
4	0.1	0.40	0.0	0.1	0.2	0.20	0.0

NAME: Toghrul Tahirov St Number: G47609664

2. Using original matrix calculate the IDF for Computer, Memory and cache showing values. Then modify original matrix using those to calculate weights.

   The below formula was used to calculate the weights:
   $WEIGHT_{ij} = TF_{ij} * [Log_2(n/IF_j) + 1]$
   Below are the python code and its results:

```python
# List of columns
cols = ["math", "physics", "computer", "cpu", "memory", "disk", "cache"]

# Define the term/document table
table = np.array([[2, 1, 0, 0, 4, 5, 8],
                  [0, 0, 0, 5, 5, 0, 0],
                  [2, 2, 2, 2, 4, 6, 2],
                  [0, 3, 0, 3, 4, 0, 0],
                  [3, 12, 0, 3, 6, 6, 0]])

total_num_docs = table.shape[0]
# Calculate the inverse document frequency for each term
if_term = np.count_nonzero(table, axis=0)

# Calculate the idf for each term
idf_term = np.log2(total_num_docs / if_term)


# for i, ift in enumerate(if_term):
#     print(f"number of documents containing the term '{cols[i]}': {ift} | IDF = {idf_term[i]}")

terms_df = pd.DataFrame(table, columns=cols)

term_metrics_dict = {'term': cols,
                     'if': if_term,
                     'idf': idf_term}
term_metrics_df = pd.DataFrame(term_metrics_dict)
# Calculate and display the weighted term frequencies
weighted_terms_df = terms_df * (idf_term + 1)
```

[8] ✓ 0.0s

NAME: Toghrul Tahirov St Number: G47609664

```
[9]  1 term_metrics_df
     ✓ 0.0s
```

	term	if	idf
0	math	3	0.736966
1	physics	4	0.321928
2	computer	1	2.321928
3	cpu	4	0.321928
4	memory	5	0.000000
5	disk	3	0.736966
6	cache	2	1.321928

+ Code + Markdown

```
[10] 1 weighted_terms_df
     ✓ 0.1s
```

	math	physics	computer	cpu	memory	disk	cache
0	3.473931	1.321928	0.000000	0.000000	4.0	8.684828	18.575425
1	0.000000	0.000000	0.000000	6.609640	5.0	0.000000	0.000000
2	3.473931	2.643856	6.643856	2.643856	4.0	10.421794	4.643856
3	0.000000	3.965784	0.000000	3.965784	4.0	0.000000	0.000000
4	5.210897	15.863137	0.000000	3.965784	6.0	10.421794	0.000000

3. Using original matrix calculate the Signal weight for physics, memory and cache showing each value. Then update matrix using those values to calculate weights.

   The signal weights formula was used as follows:

   $Weight_{ik} = TF_{ik} * [Log_2(TOTF_k) - - TF_{ik}/TOTF_k \, Log_2 (TF_{ik}/TOTF_k)]$

   Here is the Python code:

NAME: Toghrul Tahirov St Number: G47609664

```python
# List of columns
cols = ["math", "physics", "computer", "cpu", "memory", "disk", "cache"]

# Define the term/document table
table = np.array([[2, 1, 0, 0, 4, 5, 8],
                  [0, 0, 0, 5, 5, 0, 0],
                  [2, 2, 2, 2, 4, 6, 2],
                  [0, 3, 0, 3, 4, 0, 0],
                  [3, 12, 0, 3, 6, 6, 0]])

# Total occurrences of each term
total_num_terms = np.sum(table, axis=0)

# Term frequency table
term_freqs = table / total_num_terms

# Calculate the probability of a term occurring in a document
term_idfs = np.log2(term_freqs)

# Replace inf values with 0
term_idfs[np.isinf(term_idfs)] = 0

# Average information value of each term
info_val = (-1.0 * term_freqs) * term_idfs

# Average information value for each term
ave_info = np.sum(info_val, axis=0)

# Signal weights for each term
signal_weights = table * (np.log2(total_num_terms) - ave_info)

# terms_df = pd.DataFrame(table, columns=cols)

signal_weights_df = pd.DataFrame(signal_weights, columns=cols)
signal_weights_df
```

[44] ✓ 0.0s

And here is the output:

NAME: Toghrul Tahirov St Number: G47609664

```
37
```

[44] ✓ 0.0s

```
C:\Users\togru\AppData\Local\Temp\ipykernel_6028\558800666.py:20: RuntimeWarning: divide by zero encountered in log2
  term_idfs = np.log2(term_freqs)
```

</>

	math	physics	computer	cpu	memory	disk	cache
0	2.501396	2.765247	0.0	0.000000	8.890333	12.537997	20.8
1	0.000000	0.000000	0.0	8.892083	11.112916	0.000000	0.0
2	2.501396	5.530493	2.0	3.556833	8.890333	15.045597	5.2
3	0.000000	8.295740	0.0	5.335250	8.890333	0.000000	0.0
4	3.752095	33.182958	0.0	5.335250	13.335500	15.045597	0.0