# IT 458 - Information Retrieval
# Assignment 3 : Best Match Models

Jaidev Chittoria

181IT119

Date: 8 Nov 2021

**1. Original BM1, BM15, BM11**

Frequency Map A dictionary that maps each term in the document collection, to the frequency of the term in the document collection

```
In [391]: frequency_map
```

```
Out[391]: {'similitud': 8,
           'hyperson': 106,
           'real': 16,
           'flow': 430,
           'slender': 60,
           'bodi': 167,
           'blunt': 77,
           'nose': 59,
           'basi': 36,
           'small': 117,
           'perturb': 15,
           'theori': 224,
           'law': 32,
           'inviscid': 54,
           'field': 107,
           'thin': 41,
           'examin': 30,
           'restrict': 25,
           'ideal': 28,
```

**BM1**

Using the frequency map to construct the term matrix based on the BM1 formula: (N - ni + 0.5)/(ni + 0.5)

```
In [286]: term_matrix = {}
          # print(N)
          for key in sorted(frequency_map):
              x=(N - frequency_map[key] + 0.5)/(frequency_map[key] + 0.5)
              term_matrix[key] = math.log(x, 2)

          term_matrix
```

```
Out[286]: {'abbrevi': 8.844444240792825,
           'abil': 8.105384749247602,
           'abl': 7.253190908704789,
           'ablat': 7.253190908704789,
           'abovement': 8.844444240792825,
           'abrupt': 8.105384749247602,
           'abruptli': 8.105384749247602,
           'absenc': 8.105384749247602,
           'absent': 8.844444240792825,
           'absolut': 6.961581237141928,
           'absorb': 7.253190908704789,
           'absorpt': 7.617860981241781,
           'abstract': 8.105384749247602,
           'academ': 8.844444240792825,
           'acceler': 5.353280513589814,
           'accept': 6.961581237141928,
           'accommod': 8.105384749247602,
           'accompani': 6.164642333361623,
           'accomplish': 6.509906931997541,
```

Using the TF-IDF matrix constructed for the corpus in the previous assignment, we can determine whether a vocabulary term is absent (0) or present (greater than 0) in a particular document.

```
df = df.replace(np.nan, 0)
df
```

| | wing | treatment | theoret | layer | load | specif | increment | free | basi | slipstream | ... | prone | unconserv | thicker | bevel | stin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.021293 | 4.84549 | 2.511589 | 1.380604 | 3.702532 | 4.260528 | 13.69098 | 2.476256 | 4.260528 | 24.683429 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **1** | 0.000000 | 0.00000 | 0.000000 | 4.586267 | 0.000000 | 0.000000 | 0.00000 | 6.401030 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **2** | 0.000000 | 0.00000 | 0.000000 | 2.761208 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **3** | 0.000000 | 0.00000 | 0.000000 | 4.141812 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **4** | 0.000000 | 0.00000 | 0.000000 | 1.380604 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **685** | 5.432414 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 9.430453 | 9.430453 | 0.00000 |
| **686** | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **687** | 0.000000 | 0.00000 | 2.511589 | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |
| **688** | 0.000000 | 0.00000 | 0.000000 | 1.380604 | 0.000000 | 0.000000 | 0.00000 | 2.476256 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 9.43045 |
| **689** | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.00000 |

690 rows × 3279 columns

Splitting the query into composite words. Followed by using the TF-IDF dataframe and the term matrix to calculate relevance information for each document.

```
In [297]: query = "flow past"
          queryTerms = query.split(' ')
          for i in range(len(queryTerms)):
              queryTerms[i] = queryTerms[i].lower()

          queryTerms

Out[297]: ['flow', 'past']
```

```
In [298]: queryCount = dict.fromkeys(queryTerms, 0)
          for term in queryTerms:
              queryCount[term]+=1

          queryCount

Out[298]: {'flow': 1, 'past': 1}
```

Ranking the documents based on relevance

```
In [350]: ranking_dict = {}

          for i in range(N):
              ranking_dict[i] = result[i]

          ranking_dict

Out[350]: {0: 0,
           1: 3,
           2: 3,
           3: 0,
           4: 0,
```

For analysis, returning the top 3 most relevant documents, along with relevant information. The original document contents are also output for reference.

In [351]:
```
from heapq import nlargest
final_result = nlargest(3, ranking_dict, key = ranking_dict.get)
final_result
```

Out[351]:
```
[1, 2, 26]
```

In [352]:
```
print("Top 3 documents: ")
for doc in final_result:
    print("Document", doc, ": ", result[doc])
```

```
Top 3 documents:
Document 1 :  3
Document 2 :  3
Document 26 :  3
```

In [353]:
```
for doc in final_result:
    loc = 'prep_corpus/doc'+str(doc+1)
    file = open(loc, 'r')
    for line in file:
        print(line)
```

```
simpl shear flow past flat plate incompress fluid small viscos studi high speed viscou flow past two dimension bodi
usual necessari consid curv shock wave emit nose lead edg bodi consequ exist inviscid rotat flow region shock wave b
oundari layer situat aris instanc studi hyperson viscou flow past flat plate situat somewhat differ prandtl classic
boundari layer problem prandtl origin problem inviscid free stream outsid boundari layer irrot hyperson boundari lay
er problem inviscid free stream must consid rotat possibl effect vortic recent discuss ferri libbi present paper sim
pl shear flow past flat plate fluid small viscos investig shown problem treat boundari layer approxim novel featur f
ree stream constant vortic discuss restrict two dimension incompress steadi flow
boundari layer simpl shear flow past flat plate boundari layer equat present steadi incompress flow pressur gradient
newtonian flow theori slender bodi aid aerodynamieist design air frame hyperson speed speed faster mach newtonian fl
ow theori examin point view dynam hyperson small disturb theori usual theori shown result first approxim expans vali
d small basic similar paramet introduc gener solut first approxim flow past slender bodi bodi caus small disturb str
eam zero angl attack given import condit limit applic theori note name pressur coeffici surfac fall zero theori appl
i cone bodi whose shape
```

## BM15 and BM11

In [355]:
```
def term_freq_factor_Fij(term, document):
    K1 = 5
    S1 = K1+1
    fij = dictList[document][term]
    Fij = (S1 * fij)/(K1 + fij)

    return Fij
```

In [356]:
```
dictList[0]
```

Out[356]:
```
{'': 0,
 'decay': 0,
 'shockwav': 0,
 'wing': 3,
 'homann': 0,
 'grashof': 0,
 'rake': 0,
 'gasdynam': 0,
 'stall': 0,
 'advers': 0,
 'effus': 0,
 'transmiss': 0,
 'poiseuil': 0,
 'irrit': 0,
 'unit': 0,
 'refin': 0,
 'cavit': 0,
 'diagon': 0,
 'pronounc': 0,
```

In [357]:
```
term_freq_factor_Fij('face', 0)
```

Out[357]:
```
0.0
```

```
 95,
 33,
 54,
 73,
 147,
 216,
```

In [360]:
```
from statistics import mean
avg_doc_len = mean(all_doc_len)
avg_doc_len
```

Out[360]:
```
91.21014492753623
```

In [361]:
```
def doc_len_normal(term, document):

    K1 = 5
    S1 = K1+1
    fij = dictList[document][term]
    doc_length = all_doc_len[document]

    F_dash = (S1 * fij)/(((K1 * doc_length)/avg_doc_len) + fij)

    return F_dash
```

In [362]:
```
doc_len_normal('flow', 3)
```

Out[362]:
```
3.2598955410713515
```

## Correction Factor G

In [363]:
```
def correction_factor(document, query_length):
    K2 = 0
    doc_length = all_doc_len[document]
    G = K2 * query_length * ((avg_doc_len - doc_length)/(avg_doc_len + doc_length))

    return G
```

In [364]:
```
correction_factor(686, 2)
```

Out[364]: 0.0

In [365]:
```
def term_freq_within_query(term, queryCount):
    K3 = 10
    S3 = K3 + 1

    Fiq = (S3 * queryCount[term])/(K3 + queryCount[term])

    return Fiq
```

In [366]:
```
term_freq_within_query('flow', queryCount)
```

Out[366]: 1.0

## BM15

In [380]:
```
BM15 = []
for i in range(N):
    G = correction_factor(i, querylength)
    relval = 0
    for word in queryTerms:
        table_val = df.at[i, word]
        if table_val > 0:
            #BM11
            relval = relval + doc_len_normal(word, i) * term_freq_within_query(word, queryCount) * int(term_matrix[wo

            #BM15
            #relval = relval + term_freq_factor_Fij(word, i) * term_freq_within_query(word, queryCount) * term_matrix
    relval = G + relval
    BM15.append(relval)
```

In [381]:
```
BM15
```

Out[381]:
```
[0.0,
 6.950516918734853,
 9.317185508080767,
 0.0,
```

**BM11**

In [387]:
```
BM11 = []
for i in range(N):
    relval = 0
    for word in queryTerms:
        table_val = df.at[i, word]
        if table_val > 0:
            #Simpler BM15
            #relval = relval + (((K1+1)* dictList[i][word])/K1 + dictList[i][word]) * term_matrix[word]

            #Simpler BM11
            relval = relval + (((K1+1)* dictList[i][word])/((K1*all_doc_len[i])/avg_doc_len) + dictList[i][word])* in

            #BM25
            #relval += Bij(term, i) * term_matrix[word]
    relval = G + relval
    BM11.append(relval)
```

In [390]:
```
BM11
```

Out[390]:
```
[0.0,
 23.32263868065967,
 22.315089514066496,
```

**2. Simpler BM**

- K2 tends to 0, eliminating the correction factor G.
- S1 = K1 + 1, S3 = K3 + 1
- Very large K3
- Fiq is 1

$$BM1(d_j, q) \sim \sum_{k_i[q,d_j]} \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$BM15(d_j, q) \sim \sum_{k_i[q,d_j]} \frac{(K_1 + 1) f_{i,j}}{(K_1 + f_{i,j})} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$BM11(d_j, q) \sim \sum_{k_i[q,d_j]} \frac{(K_1 + 1) f_{i,j}}{\frac{K_1 \; len(d_j)}{avg\_doclen} + f_{i,j}} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

**Simpler B11**

**Relevance ranking**

```
In [390]: BM11
          7.437250293772034,
          0.0,
          0.0,
          0.0,
          0.0,
          0.0,
          0.0,
          0.0,
          0.0,
          0.0,
          0.0,
          6.818099089989889,
          0.0,
          0.0,
          0.0,
          0.0,
          5.078205833791964,
```

**Simpler BM15**

**Relevance ranking**

In [382]:
```
BM_ranking = {}

for i in range(N):
    BM_ranking[i] = BM[i]

BM_ranking
```

```
 657: 0.0,
 658: 5.890645312256253,
 659: 0.0,
 660: 0.0,
 661: 0.0,
 662: 3.4841834930105953,
 663: 0.0,
 664: 0.0,
 665: 0.0,
 666: 0.0,
 667: 0.0,
 668: 10.426655008168618,
```

**Results**

In [388]:
```
result = sorted(BM_ranking.items(), key = lambda kv:(kv[1], kv[0]))
```

In [389]:
```
result[-3:]
```

Out[389]:
```
[(105, 8.936809719154308), (2, 9.317185508080767), (668, 10.426655008168618)]
```

**3. BM25**

In [367]:
```
def Bij(term, document):
    K1 = 1
    b = 0.99
    num = (K1 + 1)* dictList[document][term]

    den = (K1 * ((1 - b) + (b*all_doc_len[document])/avg_doc_len)) + dictList[document][term]

    return (num/den)
```

$$BM25(d_j, q) \sim \sum_{k_i[q,d_j]} \mathcal{B}_{i,j} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

- Where,

$$\mathcal{B}_{i,j} = \frac{(K_1 + 1) f_{i,j}}{K_1 \left[(1 - b) + b \frac{len(d_j)}{avg\_doclen}\right] + f_{i,j}}$$

In [390]:
```
BM
```

```
 7.437250293772034,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 6.818099089989889,
 0.0,
 0.0,
 0.0,
 0.0,
 5.078205833791964,
 0.0,
 0.0,
```

In [388]:
```
result = sorted(BM_ranking.items(), key = lambda kv:(kv[1], kv[0]))
```

In [389]:
```
result[-3:]
```

Out[389]:
```
[(105, 8.936809719154308), (2, 9.317185508080767), (668, 10.426655008168618)]
```

In [ ]:
```

```