دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

Information Retrieval Course

Fall 2023

IR-HW01

Koorosh Khavari Moghaddam

koorosh.k@aut.ac.ir

koorosh.khavari.m@gmail.com

Student ID: 401131023

Contents

Preprocessing Data

Preprocessing Document Dataset

```
1  # preprocessing : tokenization, lower case, stop-word removal
2  term_doc = {}
3  doc_term = {}
4  for index, row in docs.iterrows():
5      term_doc[row['doc_id']] = []
6      tokens_list = re.split(r'\s+', re.sub(r'[^\w\s]',' ',row['document'].lower()))
7      for token in tokens_list:
8          if token in doc_term:
9              doc_term[token] += [row['doc_id']]
10         elif (((len(token)>=2) and any(not char.isdigit() for char in token)) or (len(token)>=4)) and (token not in stops):
11             doc_term[token] = [row['doc_id']]
12         else: continue
13         term_doc[row['doc_id']] += [token]
```

[6,10] All punctuations and irrelevant numbers sequences (length lower than 4) are removed from the text and the text is converted to lowercase, tokenization is performed by [6] splitting the text based on white spaces. Also, [6] all the stop words are removed from the corpus, the list of the stop words is imported as a raw text file.

Information Retrieval using Vector Space Model

Space Vector Model Information Retrival

```
1  # find most frequent terms
2  # constants
3  DIC_LENGTH = 10000
4  NUM_RELATD = 10
5  EPSILON = 1e-10
6
7  doc_term = dict(sorted(doc_term.items(), key=lambda item: len(item[1])))
8  # make dictionary of 1000 important words
9  dictionary = {term:index for index,term in enumerate(list(doc_term.keys())[0:DIC_LENGTH])}
10 # make dictionary-term tf-idf matrix for docs
11 # this operation will result the DT matrix of shape(1000,750), which is 750 doc vectors in dictionary space
12 TD = np.zeros((len(dictionary),len(term_doc)))
13 for index,(doc,terms) in enumerate(term_doc.items()):
14     doc_in_dict = {}
15     for term in terms:
16         if term in dictionary:
17             if term in doc_in_dict:
18                 doc_in_dict[term] += 1
19             else:
20                 doc_in_dict[term] = 1
21     for term in doc_in_dict.keys():
22         TD[dictionary[term],index] = (doc_in_dict[term]/len(doc_in_dict)) * np.log10(len(term_doc)/len(doc_term[term]))
```

[3,4,5] constants defined are the size of dictionary vector space, number of related retrieved documents and the epsilon value for computation stability, feel free to modify them if required.

[7] doc_term is a function of term that returns list of the docs that contains that specific term. We used term frequency across all documents to select dictionary terms. Its inverse matrix is term_doc which will give list of document ids given specific term.

[12] TD (stands for Term-Document Matrix) is the TF-IDF representation of each document inside the dictionary vector space, therefor it has the shape (10k,750).

[13:22] derivation of the TD matrix is straight forward, for each term in the document, look at the doc_term and term_doc matrices and calculate TF-IDF inside the document vector.

```python
23 # make dictionary-term tf-idf matrix for queries
24 # this operation will result the DT matrix of shape(1000,50), which is 50 query vectors in dictionary space
25 query_term = {}
26 term_query = {}
27 for index, row in queries.iterrows():
28     term_query[row['query_id']] = []
29     tokens_list = re.split(r'\s+', re.sub(r'[^\w\s]',' ',row['query'].lower()))
30     for token in tokens_list:
31         if token in dictionary:
32             if token in query_term:
33                 query_term[token] += [row['query_id']]
34             else:
35                 query_term[token] = [row['query_id']]
36             term_query[row['query_id']] += [token]
37 
38 TQ = np.zeros((len(dictionary),queries.shape[0]))
39 for index,(query,terms) in enumerate(term_query.items()):
40     query_in_dict = {}
41     for term in terms:
42         if term in query_in_dict:
43             query_in_dict[term] += 1
44         else:
45             query_in_dict[term] = 1
46     for term in query_in_dict.keys():
47         TQ[dictionary[term],index] = (query_in_dict[term]/len(query_in_dict)) * np.log10(len(term_doc)/len(doc_term[term]))
```

[25,26] query_term is a function of term that returns list of query ids related to given term, term_query is the inverse function, will give list of terms inside given query.

[27:36] derivation of query_term and term_query functions are exactly like their counterparts in the document case.

[38] TQ (stands Term_Query Matrix) is the TF-IDF representation of each query inside the dictionary vector space, therefor it has the shape (10k,50).

[39:47] derivation of the TQ matrix is straight forward, for each term in the query, look at the query _term and term_ query matrices and calculate TF-IDF using query and document functions.

```python
48 # some black magick to find 10 smallest cosine norms! result is a 50*10 matrix of all queries with appropriate doc index
49 Rank = np.argsort(np.einsum('ij,ik->kj',TD,TQ)/np.einsum('i,j->ji',1/(EPSILON+np.linalg.norm(TD,axis=0)),1/(EPSILON+np.linalg.norm(TQ,axis=0))), axis=1)[:, -NUM_RELATD:]
```

[49] all required calculations are performed as tensor products as follows:

- Performing dot product of TD and TQ tensors to form the quotient term of the cosine similarity term. The first tensor product will result in a tensor of shape (50,750) with element $a_{ij} = q_i . d_j$
- Forming multiplication of norms of each document and query pair to use as the divisor for the cosine similarity measure, the second tensor product term is simple form tensor of shape (50,750), each of its elements are $a_{ij} = \frac{1}{|q_i| \times |d_j|}$
- At the last step, perform elementwise multiplication of the previous two tensors to simply form the whole cosine similarity matrix with shape (50,750), then simply using argsort function, pick 10 of the biggest elements in each row which indicates the most relevant documents to each query and thus has the size of (50,10) as sanity check approved.

```
51 check_results(0, Rank)
✓ 0.3s

[Query]: what is the origin of COVID-19
[RELATED DOCS INDECES]: [597 278 381 557 490 748 389 668 690 637]
[RELATED DOCS IDS]: ['1abp6oom', '1vkz0b0o', '2nvk7glh', '0m5mc320', '1sq2uvur', '105q161g', '0xkz36bj', '0cq5ee1i', '2y452utz', '1j8t52yl']
TOP FIVE RESULT FOR 1'st QUERY
[1abp6oom]: Abstract Taiwan experienced a large number of severe acute respiratory syndrome (SARS) viral infections between March and July 2003; by Sep
[1vkz0b0o]: To investigate the genetic diversity, time origin, and evolutionary history of the 2019-nCoV outbreak in China and Thailand, a total of 12
[2nvk7glh]: Background: The origin of severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) is still a debatable topic. The association of the v
[0m5mc320]: The COVID-19 pandemic is a serious and global public health concern. It is now well known that COVID-19 cases may result in mild symptoms l
[1sq2uvur]: While dromedary camels are the immediate animal source of MERS coronavirus (MERS-CoV) infection, the evolutionary origin of MERS-CoV remain
[QREL]:      query_id    doc_id
0           1  005b2j4b
1           1  0chuwvg6
2           1  0t2a5500
3           1  0v34vxlb
4           1  105q161g
5           1  11edrkav
6           1  1abp6oom
7           1  1mjaycee
8           1  1sq2uvur
9           1  22fc1qly
10          1  23yi8so0
11          1  24yavi1w
12          1  25va2cvt
13          1  262bcl7h
14          1  214xxu3v
```

Information Retrieval using Binary Independence Model

Binary Independence Model

```python
1  # main Word IDF
2  def WIDF(p): return np.log10(p/(1-p))+np.log10(len(term_doc)*np.array([1/len(doc_term[term]) for term in list(dictionary.keys())]))
3  # find which words are in which documents and then sum up the idf of present terms for each doc
4  TDB = np.where(TD != 0, 1, TD)
5  TQB = np.where(TQ != 0, 1, TQ)
6  def BIM(p): return np.argsort(np.einsum("ijk,j->ik",np.einsum("ij,ik->kij",TDB,TQB),WIDF(p)), axis=1)[:, -NUM_RELATD:]
7  BIM03 = BIM(p=0.3)
8  BIM05 = BIM(p=0.5)
9  BIM07 = BIM(p=0.7)
10 # sanity check! just to be sure results are persistant
11 check_results(0, BIM03)
```

[1] WIDF (stands for word inverse document frequency matrix) will receive value of the algorithm parameter (p) and add the estimated term to the IDF matrix of the all terms, result will be a tensor of shape (10k) which is modified IDF term for each word inside the dictionary.

[4] TDB is the binary version of the TD matrix, if the term is inside the document, will assign 1 to it, otherwise 0. Its shape is (10k,750) just like the TD.

[5] TQB is the binary version of the TQ matrix, if the term is inside the query, will assign 1 to it, otherwise 0. Its shape is (10k,50) just like the TQ.

[6] all required calculations are performed as a tensor product inside BIM function as follow:

- First tensor product will receive two binary tensors TDB and TQB and then perform elementwise and on them in a specific manner, for each query vector in TQB, its elementwise and will be added to the result tensor therefor the result tensor will have the shape of (50,10k,750).
- In the second tensor product, we simply multiply each of dot blocks of the previous tensor to the modified IDF weights and will sum alongside term axis. In simple and short term, for each query, its corresponding (10k,750) binary matrix will be elementwise multiplied by the WIDF tensor (10k) shape, and then each column will be added up and the result would be a tensor of shape (750), we have 50 queries therefor final result would be of shape (50,750) which is our beloved rank tensor
- By performing argsort, we select biggest 10 elements of each row to report, therefor the output will have the shape (50,10)

```
11  check_results(0, BIM03)
 ✓ 5.0s

[Query]: what is the origin of COVID-19
[RELATED DOCS INDECES]: [153 435 328 609 597 607 425 690 353 511]
[RELATED DOCS IDS]: ['34ytd87a', '41378qru', '2zaxn6tq', '1mjaycee', '1abp6oom', '0gmtnkbh', '22fc1qly', '2y452utz', '10ecm4wi', '5pv11lfo']
TOP FIVE RESULT FOR 1'st QUERY
[34ytd87a]: In early December, pneumonia cases of unknown origin started to appear and, on the 7thof January 2020, these cases were declared to be
[41378qru]: The newly recognised coronavirus SARS-CoV-2, causative agent of coronavirus disease (COVID-19), has caused a pandemic with huge ramific
[2zaxn6tq]: Coronaviruses, seven of which are known to infect humans, can cause a spectrum of clinical presentations ranging from asymptomatic infe
[1mjaycee]: The coronavirus disease 19 (COVID-19) is a highly transmittable and pathogenic viral infection caused by severe acute respiratory syndr
[1abp6oom]: Abstract Taiwan experienced a large number of severe acute respiratory syndrome (SARS) viral infections between March and July 2003; by
[QREL]:      query_id    doc_id
0           1  005b2j4b
1           1  0chuwvg6
2           1  0t2a5500
3           1  0y34yxlb
4           1  105q161g
5           1  11edrkav
6           1  1abp6oom
7           1  1mjaycee
8           1  1sq2uvur
9           1  22fc1qly
10          1  23yi8so0
11          1  24yavi1w
12          1  25va2cvt
13          1  262bcl7h
14          1  2l4xxu3v
```

Information Retrieval using Best Match 25

Best Match 25

```
1  # parameters
2  # k_l = 1.4
3  # b = 0.75
4  l_avg = np.mean(np.array([len(terms) for terms in term_doc.values()]))
5
6  # Term-Document matrix for BM25 algorithm
7  def TDBM(b,k_l):
8      TDBM = np.zeros((len(dictionary),len(term_doc)))
9      for index,(doc,terms) in enumerate(term_doc.items()):
10         doc_in_dict = {}
11         for term in terms:
12             if term in dictionary:
13                 if term in doc_in_dict:
14                     doc_in_dict[term] += 1
15                 else:
16                     doc_in_dict[term] = 1
17         for term in doc_in_dict.keys():
18             tf = (doc_in_dict[term]/len(doc_in_dict))
19             TDBM[dictionary[term],index] = ((k_l+1)*tf*np.log10(len(term_doc)/len(doc_term[term])))/(k_l*(1-b+b*(len(doc_in_dict)/l_avg))+tf)
20     return TDBM
21 TQB = np.where(TQ != 0, 1, TQ)
22 BM2501 = np.argsort(np.einsum("ij,ik->kj",TDBM(k_l=1.4,b=0.75),TQB), axis=1)[:, -NUM_RELATD:]
23 BM2502 = np.argsort(np.einsum("ij,ik->kj",TDBM(k_l=1.6,b=0.75),TQB), axis=1)[:, -NUM_RELATD:]
24 BM2503 = np.argsort(np.einsum("ij,ik->kj",TDBM(k_l=1.8,b=0.75),TQB), axis=1)[:, -NUM_RELATD:]
```

[4] l_avg is the average length of the terms inside the dictionary

[7:19] TDBM will define the specific Term Document matrix for the BM25 algorithm provided by two parameters b and k_l. it will change the term frequency of the Term_Document matrix as given in the BM25 algorithm

[21] TQB is the binary version of the TQ matrix, if the term is inside the query, will assign 1 to it, otherwise 0. Its shape is (10k,50) just like the TQ.

[22,23,24] whole algorithm is one simple tensor product performed in each of these lines, for each pair of the document-query we simply calculated the dot product of the corresponding TDBM and TQB tensors, it will sum up the modified TD for each word that exists inside the query of interest. The result will be again the ranking tensor of shape (50,750) and by picking top 10 elements, we will achieve our goal. We perform the algorithm for three sets of parameters as followed $[[1.4, 0.75], [1.6, 0.75], [1.8, 0.75]], a_i = (k\_l_i,\ b_i)$

```
25 # sanity check! just to be sure results are persistant
26 check_results(0, BM2501)
✓ 1.1s

[Query]: what is the origin of COVID-19
[RELATED DOCS INDECES]: [748 435 425 609 557 278 389 668 690 637]
[RELATED DOCS IDS]: ['105q161g', '41378qru', '22fc1qly', '1mjaycee', '0m5mc320', '1vkz0b0o', '0xkz36bj', '0cq5ee1i', '2y452utz', '1j8t52yl']
TOP FIVE RESULT FOR 1'st QUERY
[105q161g]: A number of virological, epidemiological and ethnographic arguments suggest that COVID-19 has a zoonotic origin. The pangolin, a s
[41378qru]: The newly recognised coronavirus SARS-CoV-2, causative agent of coronavirus disease (COVID-19), has caused a pandemic with huge ra
[22fc1qly]: Coronaviruses are the well-known cause of severe respiratory, enteric and systemic infections in a wide range of hosts including m
[1mjaycee]: The coronavirus disease 19 (COVID-19) is a highly transmittable and pathogenic viral infection caused by severe acute respiratory
[0m5mc320]: The COVID-19 pandemic is a serious and global public health concern. It is now well known that COVID-19 cases may result in mild s
[QREL]:     query_id    doc_id
0           1  005b2j4b
1           1  0chuwvg6
2           1  0t2a5500
3           1  0y34yxlb
4           1  105q161g
5           1  11edrkav
6           1  1abp6oom
7           1  1mjaycee
8           1  1sq2uvur
9           1  22fc1qly
10          1  23yi8so0
11          1  24yavi1w
12          1  25va2cvt
13          1  262bcl7h
14          1  214xxu3v
```

Ranking Evaluation using P@K

P@5 and P@10 are implemented as specified by their algorithms as shown. As we can see the precision of the BM25 and Vector Space models are significantly better than the BIM model. BIM03 refers to the BIM model with the parameter p set to 0.3, also BM2501 refer to the BM25 model with the first line of the parameters specified in previous section.

```
[P@5 Results]
[VS    P@5]: 0.39
[BIM03  P@5]: 0.28
[BIM05  P@5]: 0.30
[BIM07  P@5]: 0.30
[BM2501 P@5]: 0.40
[BM2502 P@5]: 0.40
[BM2503 P@5]: 0.40

[P@10 Results]
[VS    P@10]: 0.46
[BIM03  P@10]: 0.36
[BIM05  P@10]: 0.37
[BIM07  P@10]: 0.37
[BM2501 P@10]: 0.45
[BM2502 P@10]: 0.46
[BM2503 P@10]: 0.46
```

Ranking Evaluation using MAP

Mean Average Precision metric is calculated as specified by its algorithm. The result of the models is shown. Like the P@K metric, the MAP metric show the differences between the three model and again shows the BIM model being outperformed by VS and BM25. These result are consistance with P@K.

```
[VS    MPR]: 0.46
[BIM03  MPR]: 0.36
[BIM05  MPR]: 0.37
[BIM07  MPR]: 0.37
[BM2501 MPR]: 0.45
[BM2502 MPR]: 0.46
[BM2503 MPR]: 0.46
```

Ranking Evaluation using MRR

Ranking Evaluation metric is calculated as specified by its algorithm. Like the both previous metrics, this metric also showing the gap of the performance the therefor is consistent with them.

```
[VS    MRR]: 0.52
[BIM03  MRR]: 0.41
[BIM05  MRR]: 0.45
[BIM07  MRR]: 0.46
[BM2501 MRR]: 0.56
[BM2502 MRR]: 0.57
[BM2503 MRR]: 0.57
```