Maral Dicle Maral
03.12.2021

CMPE 493 INFORMATION RETRIEVAL

ASSIGNMENT 2: A Simple Search System for Phrase and Free Text Queries

In this assignment we built a simple information retrieval system for phrase and free text queries with tf-idf and cosine similarity approach.

Index_building module is working with:

```
python3 index_build.py ./reuters21578/ ./stopwords.txt
```

As the index is built, Query_processing module is working with:

```
python3 query_process.py
```

Assignment consists of 3 main steps:

1- Data Preprocessing
2- Index Building
3- Query Processing, Phrase Search and Free Text Search

1- Data Preprocessing

Reuters data was read via an HTML file processing library, beautifulSoup. As the documents are read, they are trimmed from punctuations according to the string.punctuation list and cleaned from stop words specified by the user. Latin-1 encoding was used for the documents. Multiprocessing which creates a process for each document was used to speed up the data reading at data preprocessing step.

Maral Dicle Maral

03.12.2021

2- Index Building

I built two different indexes that will be used for phrase and free text queries. For phrase queries I used a nested dictionary structure in which words are hashed to a dictionary of documents which are hashed to positions of the corresponding word in the particular document. ({word1:{doc1:[pos1,pos2],doc2[pos1,pos2]}}) For free text queries I built three different dictionaries, one nested structure for tf_idf values of words in documents ({word1:{doc1: tf_idf1, doc2: tf_idf2}, word2……}), another dictionary structure for idf values of all the words ({word1: idf1, word2: idf2}) and one for the document norms ({doc1: norm1, doc2: norm2, doc2: norm2}) .

I preferred dictionary (hashmap structure in python) for indexes because it uses a hash function to link key and values and it provides an O(1) time complexity to store and retrieve a particular object. I preferred a basic array structure for positions list as it has O(1) complexity for adding new elements. It has O(n) complexity for searching items but as the positions lists of words are not large, I thought it won't create a problem during query search. Also, during the index building step, I didn't use any sorting function, which made the process a lot faster.

```
Last login: Tue Dec  7 16:41:04 on ttys001
maral@192 ~ % cd /Users/maral/Documents/Classes/INFO_RETRIEVAL/Assignment2_tf_idf_cosine_search_engine
maral@192 Assignment2_tf_idf_cosine_search_engine % python3 index_build.py ./reuters21578/ ./stopwords.txt
- Index is created in 7.914770 seconds -
maral@192 Assignment2_tf_idf_cosine_search_engine %
```

Figure 1 (Index Build step takes 7.914770 second for the given reuters data.)

3- Query Processing

At the last step of the assignment, query processing, a phrase query search engine and a free text query search engine is created. Query input is taken from the user and preprocessed, cleaned from stop words and punctuations before it was sent to the engines.

Phrase query search engine performs a conjunction operation in the documents and positions level by using positional index for words. Sequent words are processed dynamically (as in the

conjunction operation) by taking into account their positional information. Results are displayed in the ascending order of document IDs.

```
maral@192 Assignment2_tf_idf_cosine_search_engine % python3 query_process.py
Index is succesfully loaded!
Please enter a query (Use "w1 w2...wn" for phrase queries, w1 w2...wn for free text queries.). Enter E to exit.
Please enter a query(''E'' for exit):"James Baker"
"James Baker" is found at:
[52, 175, 190, 348, 458, 854, 965, 1357, 1392, 2052, 2078, 2190, 2452, 2620, 2632, 2633, 2648, 2701, 3421, 4008, 4038, 4139, 4764, 5176, 5186, 5290, 5294, 5759, 5869, 5944, 5964, 59
78, 6016, 6028, 6032, 6061, 6066, 6285, 6337, 6631, 6896, 7135, 7343, 7360, 7405, 7493, 7651, 7669, 7695, 7706, 7721, 7811, 8072, 8077, 8097, 8118, 8189, 8195, 8202, 8203, 8252, 830
9, 8310, 8315, 8483, 8548, 8621, 8664, 8714, 8963, 9022, 9055, 9134, 9220, 9498, 9501, 9515, 9524, 9533, 9535, 9603, 9628, 9656, 9657, 9673, 9689, 9690, 9693, 9694, 9697, 9764, 9788
, 9795, 9797, 9798, 9813, 9816, 9871, 9946, 10733, 10780, 10799, 11209, 11281, 11297, 11314, 11322, 11418, 11430, 11444, 11446, 11450, 11455, 11460, 11551, 11734, 11753, 11998, 1202
7, 12047, 12070, 12092, 12120, 12145, 12333, 12650, 12653, 12792, 12806, 12848, 13046, 13144, 13278, 13347, 13631, 13657, 13982, 14043, 14654, 14708, 14710, 14727, 14734, 14748, 147
57, 14780, 15048, 15453, 15470, 15549, 15656, 15957, 16006, 16051, 16075, 16092, 16200, 16540, 16780, 16852, 17126, 17138, 17200, 17218, 17231, 17268, 17296, 17320, 17321, 17325, 17
363, 17368, 17420, 17953, 17976, 18005, 18091, 18154, 18250, 18299, 18404, 19993, 20038, 20053, 20071, 20087, 20325, 20447, 20496, 20500, 20631, 20656, 20739, 20764, 20772, 20780, 2
0795, 20868, 20893, 20907, 20925, 21138, 21187, 21202, 21277, 21303, 21339, 21422, 21477, 21508, 21512, 21521, 21542, 21556, 21573]
- Process finished in 0.002406 seconds -
Please enter a query(''E'' for exit):
```

Figure 2 (Phrase query search for "James Baker", sorted by IDs. Process took 0.002406 second.)

Free text query search engine first performs a disjunction operation on documents for the given query. Found documents are then processed for vectorization according to the query terms. ‘0’ is entered to the document vectors for the not found words. As the vectors for the documents are ready, tf-idf values are calculated for the query as well.

Finally, query and document vectors are compared in terms of their cosine similarity. Norm_index is used here for the length normalization of the documents. Because the length of query entered and the documents in our dataset might be different, cosine similarity function might be mistaken. Therefore, by this normalization, because we are using only the unit vectors of the documents, our results are more robust. The found documents are displayed in the descending order of their cosine similarities to the query.

```
Please enter a query(''E'' for exit):turkish lira
turkish lira is found at:
[(11944, 0.3324810187453066), (12522, 0.331116600549995), (12877, 0.3272736376924795), (3076, 0.3263111415916086), (13919, 0.3225422935417205), (3518, 0.1628714560888763), (444, 0
.1509452954778111 3), (11894, 0.14700776730685636), (18061, 0.14303769625412374), (10636, 0.14174075072633205), (7105, 0.13574236723249672), (10620, 0.11981641284597595), (4328, 0.11
919502054351179), (7394, 0.11913567560281421), (10340, 0.11266856555637773), (3635, 0.11172114248205635), (5274, 0.1096519232915965), (11823, 0.10850676719857899), (1975, 0.10733392
245289723), (9831, 0.10648710741283846), (15338, 0.10499423133199311), (10862, 0.10135640149086939), (3745, 0.09622898809703634), (18288, 0.09552203427420894), (20613, 0.08927430547
354288), (2246, 0.08885894251962834), (10522, 0.08692574675984324), (10641, 0.0823315115984931), (15884, 0.07995065619086651), (13052, 0.07930658639581827), (10452, 0.07629883964804
426), (2517, 0.07594293504733281), (2970, 0.07350556069607388), (10910, 0.07211549022515791), (17300, 0.07085189446513918), (17293, 0.07079070055337815), (5574, 0.07057257902409762),
 (10627, 0.07026759691386525), (6392, 0.06925441215113065), (835, 0.06647238962567668), (10797, 0.06553484658537356), (10630, 0.0647556515250517), (9908, 0.06387181925887948), (5656
, 0.06341194575256 42), (10588, 0.06301840636337643), (10621, 0.05896019258711982), (13251, 0.05827754648483186), (15200, 0.05766036013491354), (10395, 0.05294946196022133), (1611, 0
.05277930129867623), (14664, 0.05255907450018959 5), (15534, 0.05072829680091650 6), (4809, 0.0477141824218925), (14419, 0.03799722688962528), (2522, 0.03538475597496732)]
- Process finished in 0.001641 seconds -
Please enter a query(''E'' for exit):
```

Figure 3 (Free text query search for Turkish lira, sorted by cosine similarity. First element of the tuples represents the document ID’s, second element represents the cosine similarity. Process took 0.001641 second.)