Name: Zihao ZHOU
Andrew ID: zihaozho

Machine Learning for Text Mining
Homework 1

1. Statement of Assurance

I certify that all of the material that I submit is original work that was done only by me"
Zihao ZHOU, 12/06/2020

2. Experiments

a) Describe the custom weighing scheme that you have implemented. Explain your motivation for creating this weighting scheme.

For the custom weighing scheme, I wrote a MinMax_Scaler to normalize the score from PageRank and Search Engines to 0 to 1. Then, combine both scores with a proportion of 1 to 1 to get the final score of each document. The formula is as follow:

$$IRscore(normalized) = \frac{IRscore - \min(IRscore)}{\max(IRscore) - \min(IRscore)}$$

$$PRscore(normalized) = \frac{PRscore - \min(PRscore)}{\max(PRscore) - \min(PRcore)}$$

$$score(normalized) = 0.5 * IRscore(normalized) + 0.5 * PRscore(normalized)$$

b) Report on the performance of the 9 approaches.

I. Metric: MAP

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0460	0.2588	0.2627
QTSPR	0.0461	0.2591	0.2619
PTSPR	0.0443	0.2584	0.2620

II. Metric: Precision at 11 standard recall levels

1. Precision Averages at 0.00 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.1455	0.8099	0.8319
QTSPR	0.1380	0.8036	0.8319
PTSPR	0.1305	0.8138	0.8319

2. Precision Averages at 0.10 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0877	0.5835	0.5888
QTSPR	0.0863	0.5878	0.5860
PTSPR	0.0790	0.5829	0.5874

3.Precision Averages at 0.20 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0784	0.4734	0.4728
QTSPR	0.0779	0.4662	0.4706
PTSPR	0.0673	0.4622	0.4724

4. Precision Averages at 0.30 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0740	0.3756	0.3788
QTSPR	0.0723	0.3777	0.3780
PTSPR	0.0709	0.3757	0.3782

5. Precision Averages at 0.40 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0697	0.3113	0.3142
QTSPR	0.0678	0.3136	0.3139
PTSPR	0.673	0.3117	0.3148

6. Precision Averages at 0.50 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0651	0.2418	0.2427
QTSPR	0.0625	0.2423	0.2431
PTSPR	0.0628	0.2424	0.2427

7. Precision Averages at 0.60 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0532	0.1657	0.1675
QTSPR	0.0509	0.1665	0.1672
PTSPR	0.0502	0.1661	0.1671

8. Precision Averages at 0.70 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0295	0.0915	0.0915
QTSPR	0.0271	0.0912	0.0912
PTSPR	0.0280	0.0915	0.0916

9. Precision Averages at 0.80 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0113	0.0573	0.0568
QTSPR	0.0112	0.0564	0.0555
PTSPR	0.0114	0.0550	0.0548

10. Precision Averages at 0.90 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0075	0.0388	0.0389
QTSPR	0.0074	0.0386	0.0387
PTSPR	0.0072	0.0387	0.0388

11. Precision Averages at 1.00 recall

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0040	0.0102	0.0103
QTSPR	0.0041	0.0106	0.0104
PTSPR	0.0040	0.0107	0.0104

III. Metric: Wall-clock running time in seconds

1. PageRank Time(s)

Method \ Weighting Scheme	NS	WS	CM
GPR	0.4043	0.4043	0.4043
QTSPR	6.7113	7.2375	7.9072
PTSPR	6.8782	6.7332	7.2614

PS:

1)The PageRank time of GPR is the time to reach the convergence of the iteration to get the rank of $r^{(t)}$, thus the time is the same in NS, WS, and CM. That is to say, this is the time to get 1 convergence of $r^{(t)}$.

2) The PageRank time of QTSPR and PTSPR is the sum of the PageRank time of a total of 12 topics. That is to say, we will get 12 p_t with respect to 12 topic-doc distribution and loop 12 times to get 12 $r^{(t)}$. The time shown in the table is the time used to get 12 convergences.

2. Retrieval Time(s)

Method \ Weighting Scheme	NS	WS	CM
GPR	0.1080	0.1520	0.1542
QTSPR	0.1643	0.1882	0.2127
PTSPR	0.1843	0.1824	0.2467

PS：

1)The time listed in the table is the time use of retrieval of 17885 documents in ‘indri-lists’.

IV. Parameters

For NS, I only used the $r^{(t)}$ from PageRank, convergence to rank 17885 documents in ‘indri-lists’.

For WS, after analyzing the value of PRscore and IRscore, the magnitude of PRscore(10^-9-10^5) is much smaller than IRscore's(-10-0). Therefore, the weight for PRscore is 0.999,

and IRscore is 0.001. This can approximately move these two kinds of scores to the same magnitude, which is reasonable for the combination.

For CM. I used min-max normalization for both IRscore and PRscore, which is illustrated at a) in detail.

c) Compare these 9 approaches based on the various metrics described above.

From 9 approaches above, here are some of my observations:

1) No matter which method we use, the result of the NS is useless compared with the other two methods. It seems that the information got from PageRank is not useful to rank the documents.
2) No matter which method we use, the CM is better than the other two Weighting Schemes. It shows that min-max normalization can actually increase the MAP and Precision at the same Recall level.
3) The use of the QTSPR and PTSPR method can increase the MAP and Precision compared with GPR at with same Weighting Scheme. The information got from user-topic-distribution and query-topic-distribution can increase the model ranking ability.
4) Compared with QTSPR and PTSPR, the MAP and Precision at different Recall is mostly at the same level. If we dig into the specific value of MAP and Precision at different Recall, the value of QTSPR is slightly larger than PTSPR mostly. It maybe shows that the information got from the user's topical interest distribution is a little more useful than the query's topical distribution's.

d) Analyze these various algorithms, parameters, and discuss your general observations about using PageRank algorithms.

GPR only uses information about the doc-doc relationship. From the result, we can see that it is not so useful and the precision of the result is low.

QTSPR and GTSPR both use the doc-topics information to get the PageRank rank corresponding to a specific topic(get specific p_t instead of p_0). Combined with given topic-related input, we can get a better search result rank.

QTSPR and GTSPR use a different distribution of user-query which focuses on different aspects of the query.

The search-relevance score is much more useful than PRscore. When I increase the weight of the IRscore, the MAP and Precision at a specific Recall increase.

e) 1. What could be some novel ways for search engines to estimate whether a query can benefit from v?

We can compare the results of whether using personalization or not. That is to say, if we use personalization for a specific query but the MAP of this method doesn't increase, it may show that this kind of query will not benefit from the use of personalization. With former experience, we can categorize a list of queries that are sensitive to personalization.

2. What could be some novel ways of identifying the user's interests (e.g. the user's topical interest distribution Pr(t|u)) in general?

We can use the information on existing large data in a search engine to build a neural network to estimate the probability of a user's interest in a specific topic. For example, in the user's profile, we can crawl interests from their self-introduction corresponding to the features that these users have (e.g. location, race, nationality). With enough data, we build a deep learning model to predict the possible Pr(t|u)) of a given query of users.

3. Details of the software implementation

a) **Describe your design decisions and high-level software architecture;**

My program consists of several sub-programs, which are:

1) extract_score.py
   This program will extract a search score from 'indri-lists' with respect to userid and queryid.
2) get_pt.py
   This program will get data from 'doc_topics.txt' and get p_t with respect to a different topic.
3) get_distr.py
   This program will get data from 'query-topic-distro.txt' and 'user-topic-distro.txt'and to get distribution Pr(t|u)) and Pr(t|q)).
4) trans_matrix.py
   This program will get data from 'transition.txt' and build a transition matrix.
5) PTSPR.py
   This program will do other calculations about PTSPR and output results.
6) QTSPR.py
   This program will do other calculations about QTSPR and output results.
7) GPR.py
   This program will do other calculations about GPR and output results.
8) main.py
   This program will launch the whole program.

b) **Describe major data structures and any other data structures you used for speeding up the computation of PageRank;**

I used sparse.csc_matrix to store the large sparse.

For the part of "If ni= 0 then Mij ≔ n1 for j = 1 to n", I didn't store these uniform 1/81433 to sparse.csc_matrix. Instead, because only the rows not shown in 'transition.txt' has this value along the whole column when we do matrix multiplication in iteration, we add the corresponding row with weight (1- α)*(1/81433) and sum them together and get a weight. Then, we do matrix add with weighted Unit column vector. The result of this calculation is the same as the original equation shown in the course slide.

c) **Describe any programming tools or libraries and programming environment used;**
I used package scipy and numpy for sparse matrix calculation with python3.

d) **Describe the strengths and weaknesses of your design, and any problems that your system encountered**
Strength: This program is a modularity type. Therefore, it is very easy to understand each function in each sub-program and debug-friendly.
Weaknesses: I hard-code some part of this program because of the limited time. For example, the topic number, the file names of each input are hard-coded. Instead, I should create an API for them which is more flexible.