Name: Seowoo Han
Andrew ID: seowooh

Machine Learning for Text Mining

Homework 1 - Template

1. Statement of Assurance

I certify that all of the material that I submit is the original work that was done only by me.

2. Experiments

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

The query value of the provided data is negative, and the query value obtained from my program is positive. Positive values are large, but they are difficult to distinguish because of the significant differences in values between rows. So when doing calculations with the WS method, I multiplied 0.9 to give much weight to the provided data, which is well-differentiated per query. Moreover, I multiplied the query value by 0.1 to make the total sum of weights 1. For this reason, when calculating with CM, I took the log function on the query value that I found and gave a difference to the values of each row.

b) Report of the performance of the 9 approaches.

I. Metric: MAP

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0457	0.2636	0.1310
QTSPR	0.2636	0.2636	0.2636
PTSPR	0.2636	0.2636	0.2636

II. Metric: Precision at 11 standard recall levels

(Use one table for each recall level, so totally there would be 11 tables.)

ircl_prn 0.00

Method \ Weighting Scheme	NS	WS	CM
GPR	0.1446	0.8405	0.3963
QTSPR	0.8405	0.8405	0.8405
PTSPR	0.8405	0.8405	0.8405

ircl_prn 0.10

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0875	0.5926	0.2554
QTSPR	0.5926	0.5926	0.5926

PTSPR	0.5926	0.5926	0.5926

ircl_prn 0.20

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0786	0.4732	0.2292
QTSPR	0.4732	0.4732	0.4732
PTSPR	0.4732	0.4732	0.4732

ircl_prn 0.30

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0737	0.3781	0.2021
QTSPR	0.3781	0.3781	0.3781
PTSPR	0.3780	0.3781	0.3781

ircl_prn 0.40

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0699	0.3145	0.1720
QTSPR	0.3145	0.3145	0.3145
PTSPR	0.3144	0.3145	0.3145

ircl_prn 0.50

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0653	0.2430	0.1500
QTSPR	0.2430	0.2430	0.2430
PTSPR	0.2432	0.2430	0.2430

ircl_prn 0.60

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0534	0.1677	0.1107
QTSPR	0.1677	0.1677	0.1677
PTSPR	0.1679	0.1677	0.1677

ircl_prn 0.70

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0300	0.0914	0.0632
QTSPR	0.0916	0.0915	0.0915
PTSPR	0.0915	0.0915	0.0915

ircl_prn 0.80

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0115	0.0550	0.0396
QTSPR	0.0550	0.0550	0.0550
PTSPR	0.0550	0.0550	0.0550

ircl_prn 0.90

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0074	0.0388	0.0287
QTSPR	0.0388	0.0388	0.0388
PTSPR	0.0388	0.0388	0.0388

ircl_prn 1.00

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0041	0.0101	0.0099
QTSPR	0.0101	0.0101	0.0101
PTSPR	0.0101	0.0101	0.0101

III. Metric: Wall-clock running time in seconds

PageRank running time

Method \ Weighting Scheme	NS	WS	CM
GPR		1.35764	
QTSPR		5.21853	
PTSPR		4.87009	

Retrieval running time

Method \ Weighting Scheme	NS	WS	CM
GPR	0.07792	0.07788	0.79679
QTSPR	8.16395	8.18043	7.98057
PTSPR	8.08756	7.90541	7.95461

IV. Parameters

a) Weights

Alpha = 0.8 (to fix 0.8 in the task condition.)

The sum of all weight parameters should be 1, so Beta + Gamma = 0.2.

- Beta = 0.1, Gamma = 0.1 (Beta is the coefficient of the personalized E vector and Gamma is the coefficient of the E vector used in GPR. Since the effects of E_u are not known clearly, they are set to 0.1 to give uniform weight and E vector.)

b) Error threshold = 10^{-8} (In general, float confidence intervals are$10^{-6} \sim 10^{-7}$. Therefore, the values beyond this interval can be judged not to have a significant influence on the value of a float.)

c) The M matrix was decomposed into an O matrix of 1s and a Z matrix of 0s.

$$\mu^T \cdot \mathrm{r} = \mu_O{}^T \cdot \mathrm{r} + \mu_Z{}^T \cdot \mathrm{r}$$

$$\mu^T \cdot \mathrm{r} = \mathrm{M\ matrix}$$

$$\mu_O{}^T = \mathrm{O\ matrix}$$

$$\mu_Z{}^T \cdot \mathrm{r} = \mathrm{B\ matrix}$$

- O matrix: O metrics are sparse matrices created from the data in transition.txt.
- B matrix: I can make a matrix with only zeros, but the PageRank r and dot product results in B metrics. This setup makes the calculation quick and simple.

$$\mu_Z{}^T \cdot \mathrm{r} = \begin{bmatrix} \frac{r_{i1} + r_{i2} + r_{i3} + \cdots + r_{in}}{n} \\ \frac{r_{i1} + r_{i2} + r_{i3} + \cdots + r_{in}}{n} \\ \vdots \\ \frac{r_{i1} + r_{i2} + r_{i3} + \cdots + r_{in}}{n} \end{bmatrix} = \begin{bmatrix} WS \\ WS \\ \vdots \\ WS \end{bmatrix}, where\ i = 1, 2, \cdots, m$$

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

Since GPR does not consider query or user, it is evident that performance is not as good as that of QTSPR and PTSPR.

1. MAP Comparison

QTSPR and PTSPR had the same MAP value for each weight. Therefore, MAP cannot compare the performance of QTSPR and PTSPR. GPR certainly lacks the accuracy of NS and CM weight methods when compared to QTSPR and PTSPR, but the WS weight method is the same.

2. The recall-precision analysis

In all algorithms, the precision decreases as the recall value increases. The recall-precision was similar to MAP. The accuracy of all the weight methods of QTSPR and PTSPR and the WS weight method of GPR are precisely the same or differ by 0.0005. However, the NS or CM method of GPR shows a significant difference from the accuracy of QTSPR and PTSPR. Therefore, QTSPR and PTSPR algorithms are much better than GPR.

Although the performance of QTSPR and PTSPR is similar, the analysis still shows a slight difference when the recall is extended to 50% and 60%. The accuracy of PTSPR is 0.0001 ~ 0.0002 higher. This can be thought that PTSPR performs slightly better than QTSPR as we saw in the MAP.

3. Running time analysis

PageRank running time of GPR, QTSPR, and PTSPR cannot be compared. Because QTSPR and PTSPR process 12 topics, it takes much more time than GPR. It is also difficult to compare the retrieval running time. Because GPR performs matrix calculation on one topic, QTSPR and PTSPR perform matrix calculation on 12 topics. My code takes much longer because I used some for loops to handle 12 things.

Finally, if only QTSPR and PTSPR are compared, the overall running time of PTSPR is less. Therefore, when analyzing three algorithms with three metrics, PTSPR-> QTSPR-> GPR is excellent in terms of performance and running time.

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

Using GPR is less accurate than using the QTSPR and PTSPR algorithms because it uses PageRank without considering users and queries. Therefore, we can see that the given query-topic-distro.txt. And user-topic-distro.txt are critical data for calculating PageRank. The performance of QTSPR and PTSPR was not significantly different when compared with MAP and the recall-precision, so PTSPR was calculated a little faster than QTSPR. Right now, there's no significant difference in processing time because the data isn't big enough, but when you process more data, you can see a lot more running time.

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

Each person stores the information about the search term in the matrix. You store some metrics similar to this one as cosine distance. Also, information about search terms that people search in the country where the user resides is averaged and stored in metrics.

You should set the weight appropriately for the query metrics of a specific user, the query metrics similar to the user, and the query metrics that people in the user's country search on average.

If you calculate it this way, the result can get different search results for each user, but you can search without biasing the preferences of a particular user.

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?

When people search, they don't get the right results at once. This is because the user cannot correctly determine the search terms for the desired result. So when people search for something, they do it many times.

Save the searched words several times to find common words for each search word. Instead of discarding the words that came up once in this process, they are stored together, but they give less weight than words that appear frequently. When searching multiple times, don't just consider the time zone you searched for. You can search today and do the same thing tomorrow. In other words, when collecting user search terms, you should focus on keywords as well as time zone.

Collect the words searched by the user as above. Based on these words, we infer users with similar interests as this user. The collected data can be averaged to obtain the interest distribution of the user.

3. Details of the software implementation

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

Only I made one python file (main.py). There are several functions for each algorithm and creating the result text file.

Define a file directory in the main. Use the MakeWholeTxt function to make all text files in a single text file in the indri-lists folder inside the data folder. After that, preprocess the sparse matrix to be used to obtain GPR and TSPR in the main function. Obtain GPR, QTSPR, and PTSPR in that order. QTSPR and PTSPR are obtained according to the state in the TSPR function. When a state is 0, get QTSPR by calling query-topic-distro.txt. When a state is 1, get user-topic-distro.txt to get PTSPR.

<The list of the functions>

GPR(GIPV, Zt, alpha): to compute the r vector using GPR algorithm

TSPR(Input_dir, alpha, beta, gamma): to compute the r vector using TSPR algorithm

MakeWholeTxt(): to make all text files in a single text file in the indri-lists folder inside the data folder

Definerq(MaxValue, NumOfTopic, UserId, Query, pr, IPV, state): to make Rq matrix for specific user, query. 0 state means QTSPR, 1 state means PTSPR

MakeTreeEvalFileforGPR (text_dir, IPV, method): to make a file for TreeEval using GPR algorithm. 0 method means NS weight, 1method means WS weight, 2 method means CM weight.

MakeTreeEvalFileforTSPR (text_dir, pr, IPV, state, method): to make a file for TreeEval using TSPR algorithm. 0 method means NS weight, 1method means WS weight, 2 method means CM weight.

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

I used the numpy and scipy module for handling the sparse matrix. If I treat the sparse matrix like a normal matrix, the total data size is only about 80,000 x 80,000 which uses up a lot of memory. I handled the sparse matrix using scipy module, but the memory error came out. So I used another method to solve this problem. The way I used treated data of zeros and data of ones separately. The formula has been expanded a little bit.

c) Describe any programming tools or libraries and programming environment used;

- Python IDE: Visual Studio Code
  - Environment: CPU: i7, Windows 10 & Linux 18.04(with a RTX 2080Ti), 64 bit (The running time was obtained from Linux, and the performance comparison text file was obtained from Windows 10.)
- Python version: 3.7.3
- Used libraries: numpy, math, time, scipy, os, warnings
  1. numpy: calculate array, matrix more efficiently, for using scipy module
  2. math: for using the log function

3. time: calculate computation time
4. scipy: handle and calculate the sparse matrix
5. os: gain the file directory
6. warnings: for ignoring the SparseEfficiencyWarning

d) Describe strengths and weaknesses of your design, and any problems that your system encountered

- **Strength:** Since I created functions for each feature, it's easy to adjust the weight. It also does not hard code the values (e.g. the number of topics, user, etc.) so that other data, as well as the provided data, can be processed.

- **Weakness:** The filename for output to a text file is hardcoded. Also, nothing special is done to reduce the computation speed. (It only handles out of memory problems.) Make all functions in main.py file less readable.

4. Reference

- Discussion member: Insoo Kim