**Name: Your Xin Qian**

**Andrew ID: xinq**

**Machine Learning for Text Mining**

**Homework 3 - Template**

1. **Statement of Assurance**

All of the material that I submitted are my original work done only by me.

1. **Experiments**
2. Describe the custom weighing scheme that you have implemented. Explain your motivation for creating this weighting scheme.

I implemented the custom weighting scheme **as taking the exponential** of the original PageRank score, adds it to the raw indri score. This is motived by the observation that PageRank scores are distributed over a large document collection. Each individual document has a small portion and less distinguishable from each other.

1. Report of the performance of the 9 approaches.

I. Metric: MAP

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | 0.0457 | 0.2636 | 0.2636 |
| QTSPR | 0.0432 | 0.2636 | 0.2636 |
| PTSPR | 0.0458 | 0.2636 | 0.2636 |

II. Metric: Precision at 11 standard recall levels

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | 0.1446  0.0875  0.0786  0.0737  0.0699  0.0653  0.0534  0.0300  0.0115  0.0074  0.0041 | 0.8405  0.5926  0.4732  0.3781  0.3145  0.2430  0.1677  0.0914  0.0550  0.0388  0.0101 | 0.8405  0.5926  0.4732  0.3781  0.3145  0.2430  0.1677  0.0914  0.0550  0.0388  0.0101 |
| QTSPR | 0.1291  0.0761  0.0720  0.0693  0.0653  0.0609  0.0493  0.0272  0.0115  0.0073  0.0040 | 0.8405  0.5926  0.4732  0.3781  0.3144  0.2430  0.1677  0.0914  0.0550  0.0388  0.0101 | 0.8405  0.5926  0.4732  0.3781  0.3144  0.2430  0.1677  0.0914  0.0550  0.0388  0.0101 |
| PTSPR | 0.1483  0.0813  0.0765  0.0732  0.0688  0.0616  0.0494  0.0273  0.0114  0.0072  0.0040 | 0.8405  0.5926  0.4732  0.3781  0.3145  0.2430  0.1677  0.0913  0.0550  0.0388  0.0101 | 0.8405  0.5926  0.4732  0.3781  0.3144  0.2430  0.1677  0.0914  0.0550  0.0388  0.0101 |

III. Metric: Wall-clock running time in seconds

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | 7.113 | 7.105 | 7.134 |
| QTSPR | 53.197 | 53.245 | 53.208 |
| PTSPR | 54.378 | 54.251 | 54.279 |

IV. Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | Indri:pagerank=0:10  Damping factor=0.8 | Indri:pagerank=1:10  Damping factor=0.8 | Indri:pagerank=1:exp  Damping factor=0.8 |
| QTSPR | Indri:pagerank=0:10  Alpha=0.8 Beta=0.19 Gamma=0.01 | Indri:pagerank=1:10  Alpha=0.8 Beta=0.19 Gamma=0.01 | Indri:pagerank=1:exp  Alpha=0.8 Beta=0.19 Gamma=0.01 |
| PTSPR | Indri:pagerank=0:10  Alpha=0.8 Beta=0.19 Gamma=0.01 | Indri:pagerank=1:10  Alpha=0.8 Beta=0.19 Gamma=0.01 | Indri:pagerank=1:exp  Alpha=0.8 Beta=0.19 Gamma=0.01 |

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

MAP score: NS < WS = CM, GPR~=QTSPR~=PTSPR

Precision at 1st recall level: NS<<WS=CM, GPR~=QTSPR~=PTSPR

Precision at subsequent recall level: NS degrades slowly while WS and CM degrades drastically

Wall-clock running time: NS~=WS~=CM, GPR<<QTSPR~=PTSPR

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

For MAP, in NS scenario, PTSPR achieves best result given, which is possibly because it could best recover users’ information need. QTSPR has worst result given NS scenario, completely ignorant of user’s expectation. In general, NS is the weakest method as compared to WS and CM since there’s a lot of term frequency, collection-wide statistics and smoothing in indri that improves the retrieval performance, but NS does not incorporate any of the information.

We took an additional test on the original Indri result only, the MAP is 0.2635. WS improves the original Indri MAP by 0.0001. Results from GPR, QTSPR and PTSPR are mostly the same. It might because indri is already a strong and sufficient retrieval model on the corpus CiteEval. Link analysis might be a supplementary source of information on this corpus. PageRank might be more effective given another dataset, e.g. ClueWeb09, ClueWeb12, etc.

If we take a closer look at precision at each recall level, we found precision at first recall level is high for WS and CM, but at subsequent recall level they degrade drastically. This might be because the retrieval depth, we are returning top 500 documents. 500 is the retrieval depth. This retrieval depth might be much larger than the complete relevant document sets. For some queries, there might have only ~10-20 relevant documents. Therefore it is possible that even when the MAP value is small 0.2636, the retrieval performance might already be good.

Running time for GPS is around 1/8 of the QTSPR and PTSPR. It is because there’s no underlying topic assumption in GPS, power iteration is done only once. But for QTSPR and PTSPR, we need to do power iteration for all 12 topics. It is interesting why the value is not 1/12 but 1/8. I would argue it is because the imbalanced personalization vector gives faster convergence than the uniform jumping vector p.

In all three PageRank algorithm, the dampening factor is 0.8, on the belief that the link structure in the graph is main source of evidence of page importance. Prior knowledge such as query-topic distribution or user-topic distribution serves mainly as supplementary factors. We set beta as 0.19 and gamma as 0.01 on the belief that a prior knowledge-based jumping vector would be more effective than a random uniform jumping vector.

In conclusion, using PageRank might improve performance on retrieval. This argument is yet still domain-specific. For academic publication retrieval like CiteEval, adding a PageRank score to the retrieval model with some weight might be a good choice. For other domain when the graph structure is sparse, PageRank might be less effection. But to make it effective, it must be combined with other retrieval models, such as Indri. PageRank algorithm itself alone is less effective.

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

* Analyzing query logs (search trails such as click through rates, stay time in a site, etc. ) to see whether the user has been satisfied with current retrieval results.

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?

1. **Details of the software implementation**
2. Describe your design decisions and high-level software architecture;

Python’s strong support in Matrix computation and easy modularity allows me to develop the algorithm in a rapid cycle. There are

1. readTransitionMatrix.py constructs the transition matrix that could be used by all three PageRank algorithm.
2. topicSpecificTeleportationVector.py constructs the document-topic matrix
3. powerIteration takes M, p\_0 and p\_t and constructs the initial vector r to do power iteration
4. GPR.py/QTSPR.py/PTSPR.py chains the modules together, writes the PageRank score to disk
5. rerank.py takes PageRank score and parses Indri scores from disk, combine them in some way to get the final ranking

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

I am using Python’s sparse matrix to form the transition matrix (without the normalization cells). Decomposing the power iteration, I found collecting weights from the dangling nodes and reassign equally to all nodes would speed up computation than doing power iteration on the original M transition matrix defined in course slides. Since QTSPR and PTSPR both computes an arithmetic average over each topic’s query likelihood \* r. r is a shared vector over different queries. We cache r for each topic. PageRank is an algorithm that can be optimized. For GPR on average 155 iterations till convergence will cost less than 8 seconds.

1. Describe any programming tools or libraries that you used;

I’m using Python’s built-in libraries such as time, sys for counting runtime and file system IO. Additionally, I’m using scipy.ss to rank PageRank score, as long as the rank does not change, the iteration converges. I’m also using scipy for sparse matrix multiplication in power iteration.

1. Describe strengths and weaknesses of your design, and any problems that your system encountered

I utilized the handy Python library to develop code fast. This design has high modularity and readability. The weakness might be due to time limit, I didn’t alter the code to take in alpha, beta and gamma value as arguments but hard coded them in to the module. There’s no major problem encountered in this assignment.

1. **Describe how to run your code (programming environment, command line, etc.)**

The program was run on Python 2.7x. For detailed commands, please refer to readme.