**Name: Leonardo Neves**

**Andrew ID: lneves**

**Machine Learning for Text Mining**

**Homework 3**

1. **Statement of Assurance**

I assure that I am responsible for all the code and answers for this homework.

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

My custom weighting method started as the weighted harmonic mean as I thought it would have a similar result as the weighted sum, but giving a higher effect to lower scores. Thus, I realized that the page ranks and the relevance scores were not on the same order of magnitude and, to better weight them, I have taken the log of their absolute value. I got the minus because the relevance scores are negative.

1. Report of the performance of the 9 approaches.

I. Metric: MAP

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | 0.0460 | 0.2642 | 0.2637 |
| QTSPR | 0.0456 | 0.2644 | 0.2637 |
| PTSPR | 0.0469 | 0.2644 | 0.2638 |

II. Metric: Precision at 11 standard recall levels

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | ircl\_prn.0.00 all 0.1447 ircl\_prn.0.10 all 0.0874 ircl\_prn.0.20 all 0.0787 ircl\_prn.0.30 all 0.0738 ircl\_prn.0.40 all 0.0701 ircl\_prn.0.50 all 0.0655 ircl\_prn.0.60 all 0.0535 ircl\_prn.0.70 all 0.0301 ircl\_prn.0.80 all 0.0115 ircl\_prn.0.90 all 0.0074 ircl\_prn.1.00 all 0.0041  P5 all 0.0368 P10 all 0.0447 P15 all 0.0474 P20 all 0.0474 P30 all 0.0421 P100 all 0.0450 P200 all 0.0518 P500 all 0.0601 P1000 all 0.0301 | ircl\_prn.0.00 all 0.8417 ircl\_prn.0.10 all 0.5996 ircl\_prn.0.20 all 0.4757 ircl\_prn.0.30 all 0.3765 ircl\_prn.0.40 all 0.3134 ircl\_prn.0.50 all 0.2411 ircl\_prn.0.60 all 0.1688 ircl\_prn.0.70 all 0.0921 ircl\_prn.0.80 all 0.0541 ircl\_prn.0.90 all 0.0390 ircl\_prn.1.00 all 0.0098  P5 all 0.6000 P10 all 0.5105 P15 all 0.4509 P20 all 0.4145 P30 all 0.3596 P100 all 0.1939 P200 all 0.1275 P500 all 0.0601 P1000 all 0.0301 | ircl\_prn.0.00 all 0.8409 ircl\_prn.0.10 all 0.5929 ircl\_prn.0.20 all 0.4737 ircl\_prn.0.30 all 0.3789 ircl\_prn.0.40 all 0.3143 ircl\_prn.0.50 all 0.2427 ircl\_prn.0.60 all 0.1675 ircl\_prn.0.70 all 0.0916 ircl\_prn.0.80 all 0.0567 ircl\_prn.0.90 all 0.0390 ircl\_prn.1.00 all 0.0103  P5 all 0.6000 P10 all 0.5132 P15 all 0.4456 P20 all 0.4132 P30 all 0.3579 P100 all 0.1942 P200 all 0.1274 P500 all 0.0601 P1000 all 0.0301 |
| QTSPR | ircl\_prn.0.00 all 0.1444 ircl\_prn.0.10 all 0.0867 ircl\_prn.0.20 all 0.0779 ircl\_prn.0.30 all 0.0733 ircl\_prn.0.40 all 0.0695 ircl\_prn.0.50 all 0.0650 ircl\_prn.0.60 all 0.0533 ircl\_prn.0.70 all 0.0302 ircl\_prn.0.80 all 0.0114 ircl\_prn.0.90 all 0.0073 ircl\_prn.1.00 all 0.0040  P5 all 0.0368 P10 all 0.0447 P15 all 0.0491 P20 all 0.0474 P30 all 0.0421 P100 all 0.0453 P200 all 0.0517 P500 all 0.0601 P1000 all 0.0301 | ircl\_prn.0.00 all 0.8417 ircl\_prn.0.10 all 0.5994 ircl\_prn.0.20 all 0.4756 ircl\_prn.0.30 all 0.3773 ircl\_prn.0.40 all 0.3148 ircl\_prn.0.50 all 0.2417 ircl\_prn.0.60 all 0.1689 ircl\_prn.0.70 all 0.0920 ircl\_prn.0.80 all 0.0542 ircl\_prn.0.90 all 0.0391 ircl\_prn.1.00 all 0.0098 P5 all 0.6000 P10 all 0.5105 P15 all 0.4509 P20 all 0.4145 P30 all 0.3596 P100 all 0.1939 P200 all 0.1274 P500 all 0.0601 P1000 all 0.0301 | ircl\_prn.0.00 all 0.8402 ircl\_prn.0.10 all 0.5928 ircl\_prn.0.20 all 0.4735 ircl\_prn.0.30 all 0.3785 ircl\_prn.0.40 all 0.3146 ircl\_prn.0.50 all 0.2428 ircl\_prn.0.60 all 0.1674 ircl\_prn.0.70 all 0.0916 ircl\_prn.0.80 all 0.0567 ircl\_prn.0.90 all 0.0388 ircl\_prn.1.00 all 0.0103  P5 all 0.6000 P10 all 0.5132 P15 all 0.4456 P20 all 0.4145 P30 all 0.3579 P100 all 0.1937 P200 all 0.1278 P500 all 0.0601 P1000 all 0.0301 |
| PTSPR | rcl\_prn.0.00 all 0.1451 ircl\_prn.0.10 all 0.0832 ircl\_prn.0.20 all 0.0785 ircl\_prn.0.30 all 0.0744 ircl\_prn.0.40 all 0.0700 ircl\_prn.0.50 all 0.0617 ircl\_prn.0.60 all 0.0491 ircl\_prn.0.70 all 0.0270 ircl\_prn.0.80 all 0.0114 ircl\_prn.0.90 all 0.0072 ircl\_prn.1.00 all 0.0041  P5 all 0.0263 P10 all 0.0421 P15 all 0.0491 P20 all 0.0474 P30 all 0.0456 P100 all 0.0497 P200 all 0.0580 P500 all 0.0601 P1000 all 0.0301 | ircl\_prn.0.00 all 0.8417 ircl\_prn.0.10 all 0.6004 ircl\_prn.0.20 all 0.4760 ircl\_prn.0.30 all 0.3774 ircl\_prn.0.40 all 0.3144 ircl\_prn.0.50 all 0.2419 ircl\_prn.0.60 all 0.1684 ircl\_prn.0.70 all 0.0921 ircl\_prn.0.80 all 0.0543 ircl\_prn.0.90 all 0.0390 ircl\_prn.1.00 all 0.0099  P5 all 0.6000 P10 all 0.5132 P15 all 0.4526 P20 all 0.4145 P30 all 0.3605 P100 all 0.1939 P200 all 0.1276 P500 all 0.0601 P1000 all 0.0301 | ircl\_prn.0.00 all 0.8403 ircl\_prn.0.10 all 0.5935 ircl\_prn.0.20 all 0.4739 ircl\_prn.0.30 all 0.3787 ircl\_prn.0.40 all 0.3145 ircl\_prn.0.50 all 0.2425 ircl\_prn.0.60 all 0.1675 ircl\_prn.0.70 all 0.0916 ircl\_prn.0.80 all 0.0567 ircl\_prn.0.90 all 0.0388 ircl\_prn.1.00 all 0.0104  P5 all 0.6000 P10 all 0.5132 P15 all 0.4456 P20 all 0.4145 P30 all 0.3579 P100 all 0.1942 P200 all 0.1275 P500 all 0.0601 P1000 all 0.0301 |

III. Metric: Wall-clock running time in seconds

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ Weighting Scheme | NS | WS | CM |
| GPR | 0. 2117162315 | 0.1956176820 | 0.1972206078 |
| QTSPR | 0.2232007290 | 0.2210894509 | 0.2128124237 |
| PTSPR | 0.2002020195 | 0.2124132419 | 0.2261456753 |

IV. Parameters

For the GPR, I have used PRW of 1000 and relevance of 0.5 for the WS and PRW of -8 and relevanceW of 100 for the CM.

For the QTSPR, I have used beta as 0.02 and gamma as 0.18 and the same values for WS and PRW.

For PTSPR, I have used beta = 1.0 and gamma = 0 for the NS and beta = 0.04 and gamma as 0.16 for WS and CM. Same values for the weights of WS and PRW for the GPR were used.

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

All experiments were based on PR after 10 iterations.We can see that the No Search approaches have a worse result. This makes clear that the PageRank alone is not a really good metric for ranking documents. On the other hand, the search relevance alone has a baseline of 0.2636 MAP and all the results that combine the PR and the SearchRelevance data are better than that, showing that PR is able to improve performance when combined.

The query-based page rank did not have a much better result. Without the search, it shows a worse result as we could expect, since there is no real query being searched, while it got a better result for the Weighted Sum approach. The best results were the ones that used the personalized TSPR, getting higher precision and recall using the weighted sum. The Custom Method, a adaptation of the weighted harmonic mean, got a better precision on higher ranks than the weighted sum on most experiments, maybe making it more suitable for web search engines where the top ranks are more important, despite of having a MAP.

Because of the well-structured design and by caching the offline PageRank computation, we did not see much difference among computation time of any of the algorithms, all of them being fast enough to be used by a user.

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

As we could see on question C, PageRank is not a good metric alone or, at least, is not better than using the search relevance alone, but it might be used to improve the results. Personalized Topic Sensitive PageRank improved the results significantly and, even without the search relevance it showed better than the other two. For the parameters, since the PR value is much lower than the search relevance value, we had to tune the weights in order to get a better result and that could get them to be in the same order of magnitude. Also, using PTSPR asked for higher parameters for beta, showing that the teleportation matrix is more important for this configuration than for QTSPR.

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

We could consider the query to be a really small document. Also, we could pre-process documents of the same topic, maybe using the kmeans algorithm, to get a good vector to represent the given topic. Then, we could compute the similarity between the query and this representation and, if some vectors showed a high similarity to the query, it might indicate that personalization would be a good idea.

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?

One obvious but not so effective way would be to have a set of labeled documents and explicitly ask user for feedback. This has the pro of being 100% sure about the user interest but most users would not like to have those questions often. A better approach might be to monitor the user queries. If his searches frequently retrieve documents labeled as a topic, we might think he is interested on that particular topic. Also, if he searches for documents, selects one an stays a reasonable time reading that document, it might represent interest as well. This time spent and the amount of documents of a particular topic normally retrieved by the user might be good for estimating initial probabilities that could be tuned by treating this problem as a classification problem. We could them, from the assumptions, use our knowledge to rank the documents differently and observe the user response to that. If we retrieve a document based on our assumptions and the user do spend a reasonable time reading that document, we can assume it was a positive prediction and assume it was bad otherwise.

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

I have created a function to use the Power method for computing the offline matrix of pagerank. My transition matrix was created using a sparse matrix from Scipy. I have used a map of maps to map every user to every query and every user-query pair to the topics distribution. I have also used a map to store the pagerank matrices based on the topic.

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

Page rank computation was fast because I have used Sparse matrix to create the M matrix and and the normalize function from sklearn to normalize the rows by the out edges values. Sparse matrix multiplication is very fast using those libraries.

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

I have used Scipy for sparse matrix computations, Numpy for handling data and sklearn for normalizing the rows.

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

I think my design allows me to make computation very fast. It is easy to see on the experiments that the computation is so fast that it is hard to determine which method is faster. I have also cached the offline page rank computation, which made my code faster for avoiding unnecessary computation.

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

To run the code, run *pagerank.py method path-to-distro alpha beta-factor pagerankWeight searchRelevanceWeight*

Method = “WS”, “CM” or “NS”

Path-to-distro = “query-topic-distro.txt” or “user-topic-distro.txt”

Alpha = dampening factor = 0.8

Beta-factor = proportion of the remaining value (1-alpha) = 0.2 that should go to beta. The rest will go to gamma.

PageRankWeight and searchRelevanceWeight are based for the WS or CM methods.

Code assumes the indri-lists folder is in the same folder as the code.