Name: YongKyung Oh
Andrew ID: yongkyuo

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.

YongKyung Oh

Feb. 06. 2020

2. Experiments

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

First of all, I tested with the baseline data (which is the original indri-list data) and totally random weight (which is generated by uniform Dirichlet distribution).
Second, I checked the NS scheme for the comparison. Most of the performance measure (especially MAP) is lower than baseline and random score. So, I assume that our pagerank data is not sufficient to improve the over retrieval performance.
Third, I tested multiple weighted sum ratio between pagerank and search score. When increasing pagerank ratio, performance measure (MAP) decreases.
Fourth, I compare the unit scale of pagerank score and search score data. Most of all, absolute value of retrieval data is much larger than pagerank score and the signs are opposite. (retrieval data is negative and pagerank data is positive).

Therefore, I assume that the linear combination of two data is not a good strategy for this case. Instead of that I use the MinMax standardization for pagerank score and retrieval score. I tried multiple approach to generate new custom measure to improve the performance. Also, Retrieval score is more important than pagerank. For example, in the case of WS, if I increase the pagerank ratio, than the performance measures (MAP) decreases. Based on this understanding, I use the retrieval data first and add normalized custom score.

I change the p_weight ratio in the weighted sum scheme as follow. For simplicity, I only use the GPR and compare the result using MAP. Results in the figure 1.

```
method = 'GPR'
score_method = 'WS'
WS_set = [0.0, 0.00001, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 0.999, 0.9999, 0.99999]
```

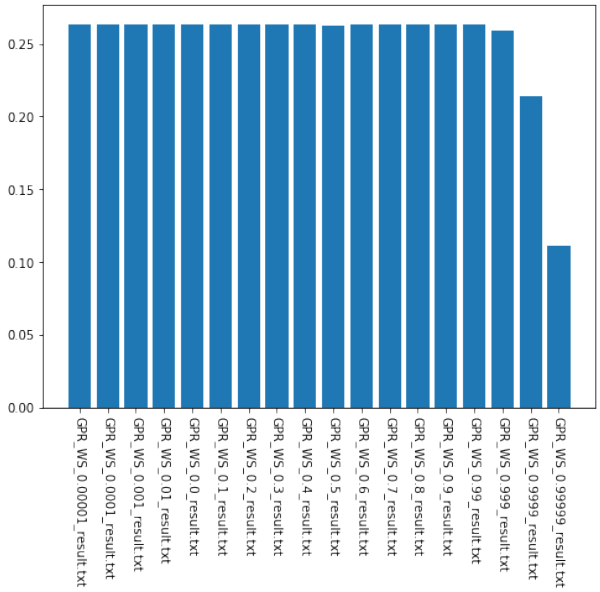


Figure. 1. Performance Comparison (MAP): Weighted sum scheme

In the custom method, I use the standardization and normalization to scale two different scores. Detail process is explained as follow.

$$\mathrm{PR} = [pagerank\ score]$$
$$\mathrm{SR} = [search\ score]$$

Pagerank and Search score have different scale and range. In order to utilize the information, I standardized the data.

$$\mathrm{Standardized}(\mathrm{x_i}) = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Therefore, minimum of each value is 0 and maximum of each value is 1. To avoid conflict in calculation, I ignore the invalid value (ex. Infinity) and substitute 0 for threshold value (1e-10).

I tested following custom score method

$$\mathrm{PR_s} = [stamdardized\ pagerank\ score]$$
$$\mathrm{SR_s} = [stamdardized\ search\ score]$$

$$\mathrm{Custom\ score} = \mathrm{SR_s} + [Normalized\ custom\ coefficient]$$

Basic assumption is that both scores is important for the performance, but the experiment results was not.

Custom coefficient is calculated by following equations.

#CM00.

$$(\text{Custom coefficient})_{01} = Dirichlet\ Random$$

#CM01.

$$(\text{Custom coefficient})_{01} = abs(\text{PR}_\text{s} + \text{SR}_\text{s})$$

#CM02.

$$(\text{Custom coefficient})_{02} = abs(\text{PR}_\text{s} - \text{SR}_\text{s})$$

#CM03.

$$(\text{Custom coefficient})_{03} = mean(\text{PR}_\text{s}, \text{SR}_\text{s})$$

#CM04.

$$(\text{Custom coefficient})_{04} = \max(\text{PR}_\text{s}, \text{SR}_\text{s})$$

#CM05.

$$(\text{Custom coefficient})_{05} = geometric\_mean(\text{PR}_\text{s}, \text{SR}_\text{s})$$

#CM06.

$$(\text{Custom coefficient})_{06} = harmonic\_mean(\text{PR}_\text{s}, \text{SR}_\text{s})$$

#CM07.

$$(\text{Custom coefficient})_{07} = -e^{\text{PR}_\text{s}} + e^{\text{SR}_\text{s}}$$

#CM08.

$$(\text{Custom coefficient})_{07} = e^{\text{PR}_\text{s}} + e^{\text{SR}_\text{s}}$$

#CM09.

$$(\text{Custom coefficient})_{09} = -\log(\text{PR}_\text{s}) + \log(\text{SR}_\text{s})$$

#CM10.

$$(\text{Custom coefficient})_{10} = -\log(\text{PR}_\text{s}) - \log(\text{SR}_\text{s})$$

So, instead of using the custom coefficient, normalized custom coefficient is added to the search score.

$$\text{Normalization}(\text{x}_\text{i}) = \frac{x_i}{sum(x)}$$

For example, in the case of #CM01, final custom score is as follow

$$\text{Custom score} = \text{SR}_\text{s} + [Normalized\ custom\ coefficient]_{01}$$

where

$$[(Normalized\ custom\ coefficient)_i]_{01} = \frac{[(Normalized\ custom\ coefficient)_i]_{01}}{[Normalized\ custom\ coefficient]_{01}}$$

Therefore, sum of the custom coefficients is equal to 1 and each coefficient is between 0 and 1. I assume that this coefficient perturbate the standardized search score. In the figure 2, the result of custom methods with regard to MAP.

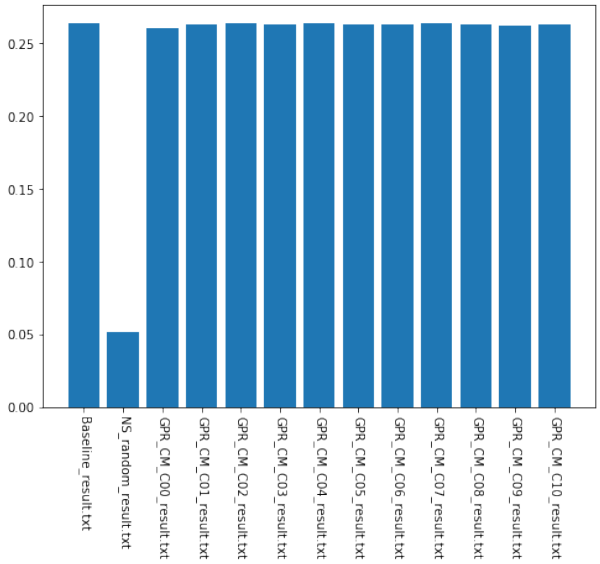


Figure. 2. Performance comparison (MAP): Custom methods

As you can see, there is no big difference between each approach. Because I keep the search score is more dominant affect the final performance. If I only use the custom coefficient, performance is poor.

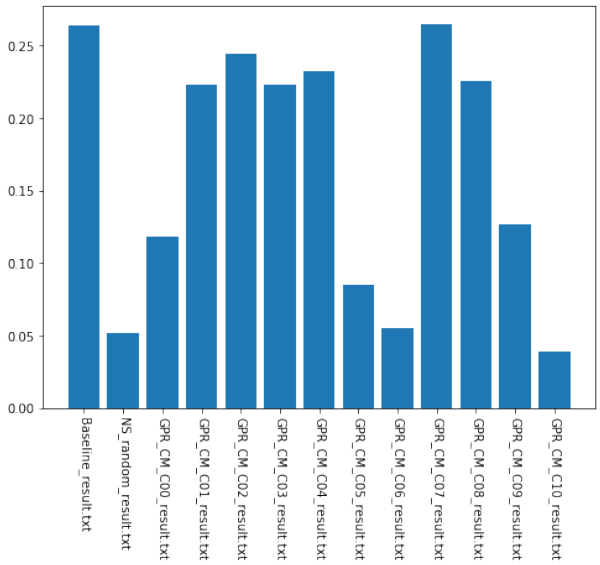


Figure. 3. Performance comparison (MAP): Custom coefficients

Also, It is interesting that #CM02 is better than #CM01 when the gap between two scores is used. Instead of using less logical approach, I selected #CM01 which is sum of standardized scores. #CM07 looks better, which is more affected by search scores. In the following section, #CM07 used CM methods.

b) Report of the performance of the 9 approaches.

I. Metric: MAP

For baseline, original indri-list MAP is 0.2635

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0454	0.2588	0.2637
QTSPR	0.0433	0.2559	0.2637
PTSPR	0.0455	0.2599	0.2637

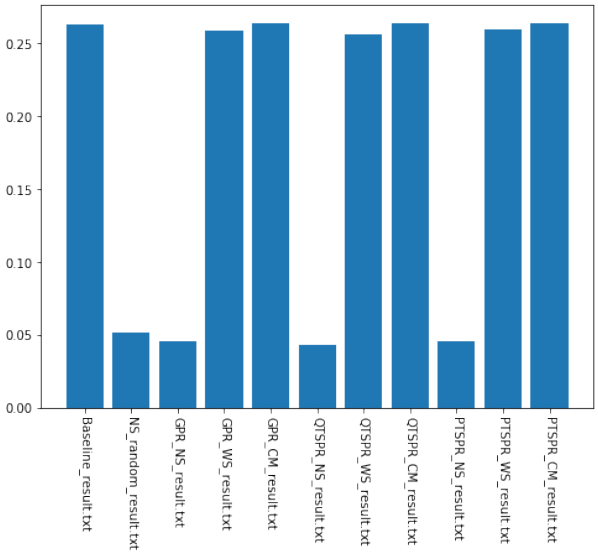


Figure. 4. Performance comparison (MAP): 9 approaches

II. Metric: Precision at 11 standard recall levels

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

Recall 0.00

Method \ Weighting Scheme	NS	WS	CM
GPR	0.1440	0.8099	0.8405
QTSPR	0.1367	0.7881	0.8405
PTSPR	0.1479	0.8124	0.8405

Recall 0.10

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0873	0.5835	0.5926
QTSPR	0.0780	0.5706	0.5926
PTSPR	0.0817	0.5870	0.5926

Recall 0.20

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0781	0.4726	0.4732
QTSPR	0.0729	0.4669	0.4731
PTSPR	0.0768	0.4710	0.4731

Recall 0.30

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0731	0.3760	0.3782
QTSPR	0.0703	0.3740	0.3781
PTSPR	0.0731	0.3780	0.3780

Recall 0.40

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0695	0.3111	0.3146
QTSPR	0.0661	0.3092	0.3146
PTSPR	0.0688	0.3128	0.3148

Recall 0.50

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0650	0.2418	0.2435
QTSPR	0.0615	0.2407	0.2435
PTSPR	0.0639	0.2409	0.2430

Recall 0.60

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0531	0.1658	0.1682
QTSPR	0.0498	0.1662	0.1682
PTSPR	0.0492	0.1671	0.1677

Recall 0.70

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0300	0.0915	0.0915
QTSPR	0.0276	0.0914	0.0915
PTSPR	0.0273	0.0912	0.0915

Recall 0.80

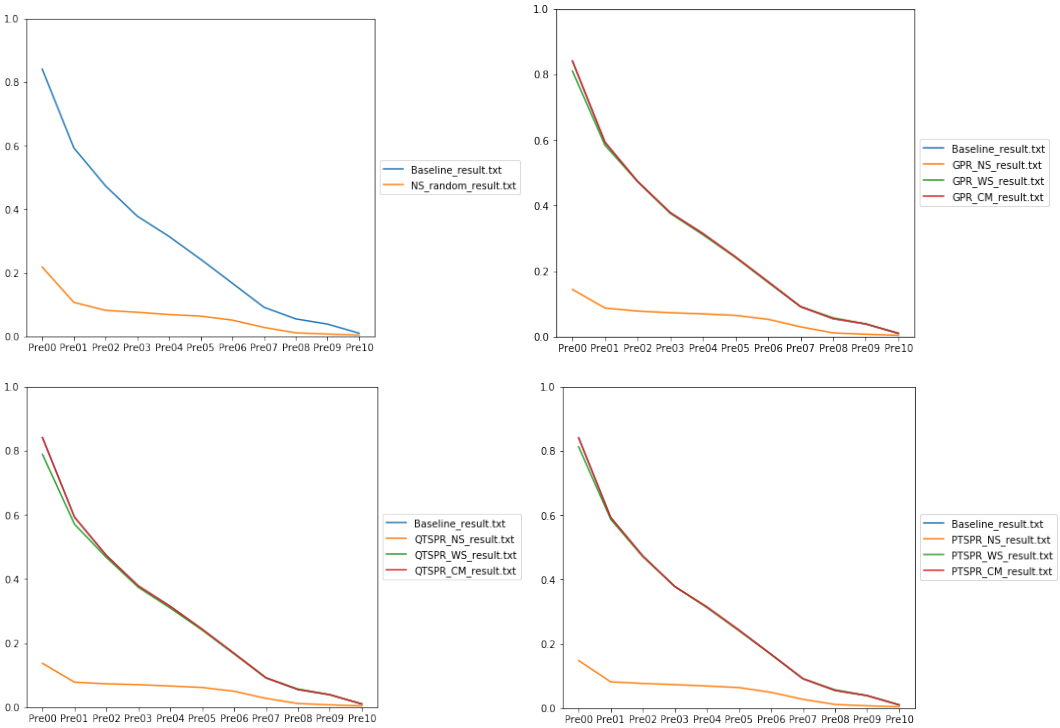
Method \ Weighting Scheme	NS	WS	CM
GPR	0.0115	0.0573	0.0550
QTSPR	0.0115	0.0568	0.0550
PTSPR	0.0114	0.0569	0.0550

Recall 0.90

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0074	0.0388	0.0388
QTSPR	0.0073	0.0392	0.0388
PTSPR	0.0073	0.0392	0.0388

Recall 1.00

Method \ Weighting Scheme	NS	WS	CM
GPR	0.0041	0.0102	0.0101
QTSPR	0.0040	0.0101	0.0101
PTSPR	0.0040	0.0102	0.0101



III. Metric: Wall-clock running time in seconds

In case of running pagerank and retrieval

Method \ Weighting Scheme	NS	WS	CM
GPR	5.2481971	5.1582354	6.1385552
QTSPR	19.1146624	21.0127382	17.2880463
PTSPR	19.4136085	20.5028115	20.567261

In case of importing pagerank data and running retrieval

Method \ Weighting Scheme	NS	WS	CM
GPR	2.0109212	1.3404788	1.6985722
QTSPR	8.8867592	7.3427424	6.7365879
PTSPR	6.6504936	7.5410225	6.6207727

Pagerank Calculation

GPR: 4.4922933 (s)

QTSPR: 17.9502158 (s)

PTSPR: 18.5707649 (s)

Pagerank Calculation + Save (json file)

GPR: 5.1626036 (s)

QTSPR: 26.8912887 (s)

PTSPR: 29.2392866 (s)

IV. Parameters

- Doc size: 81433
- Topic size: 12
- Query size: 6
- Convergence Threshold: 1e-10
- For pagerank
  - alpha (α) = 0.2
  - dampening (damping) factor: $1 - \alpha = 0.8$
  - beta (β) = 0.2
- For weighted sum
  - p_weight = 0.999
  - s_weight = 0.001

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

In all case, 'NS' is not good option. Because offline pagerank score doesn't have enough information to reflect the information retrieval of search score. Therefore, weight sum and custom method performances are lower than baseline (indri list original)

Interestingly, GPR is slight better than other results, which means our data is not good enough to improve the performance. ('query-topic-distro.txt' and 'user-topic-distro.txt')

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

Most of all, pagerank algorithm and the performance is depending on the data itself. If we have enough and almost real-time data regarding users, then the performance will be increased. At the same time, it require large scale computation for getting the result. The transition matrix is

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

Nowadays, we can simply get personalize data from users. Therefore, the importance point is how to connect the personalized insight to the query data. We can cluster the similar users / query or other data. For example, collaborative filtering technique can be combined with pagerank and search score for information retrieval.

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 implement the neural network and Bayesian inference for the probability distribution. If we have enough data to train the model, we can expect the user's interest using the model. In general neural network approach is good for finding the patterns, so we can implement this approach with other text mining techniques. Additional to that, if we have enough resource for that, we can predict specific user's interest using similar user's interest. It can be use to micro-target strategy for companies.

3. Details of the software implementation

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

pagerank_function.py

- Include pagerank function
- To get transition matrix and calculate pagerank scores: GPR/QTSPR/PTSPR

pagerank.py

- Run pagerank process using pagerank method: GPR/QTSPR/PTSPR
- User can define the method and parameters
- Enable to save the pagerank data into json format

```
$ python pagerank.py -h
usage: pagerank.py [-h] --Method METHOD [--Save SAVE] [--Output OUTPUT]
                   [--alpha ALPHA] [--beta BETA]

Python module for Pagerank Score

optional arguments:
  -h, --help            show this help message and exit
  --Method METHOD, -M METHOD
                        Pagerank methods: GPR/QTSPR/PTSPR
  --Save SAVE           Determine json output (default: False)
  --Output OUTPUT, -O OUTPUT
                        Output file name (default: [Method]_output.json
  --alpha ALPHA         Damping parameter (default: 1-alpha = 0.8
  --beta BETA           TSPR parameter (default: beta = 0.1
```

retrieval.py

- Calculate the retrieval score using score method: NS/WS/CM
- User can define the method and parameters
- Enable to save the retrieval score into txt format

```
$ python retrieval.py -h
usage: retrieval.py [-h] --Method METHOD --Score SCORE [--Save SAVE]
                    [--Output OUTPUT] [--alpha ALPHA] [--beta BETA]
                    [--p_weight P_WEIGHT] [--s_weight S_WEIGHT]

Python module for Pagerank Score

optional arguments:
  -h, --help            show this help message and exit
  --Method METHOD, -M METHOD
                        Pagerank methods: GPR/QTSPR/PTSPR
  --Score SCORE, -S SCORE
                        Retrieval score methods: NS/WS/CM
  --Save SAVE           Determine json output (default: False)
  --Output OUTPUT, -O OUTPUT
                        Output file name (default:
                        [Method]_[Score_method]_output.txt
  --alpha ALPHA         Damping parameter (default: 1-alpha = 0.8
  --beta BETA           TSPR parameter (default: beta = 0.1
  --p_weight P_WEIGHT   WS method parameter (default: p_score = 0.999
  --s_weight S_WEIGHT   WS method parameter (default: s_score = 0.001
```

pagerank_sample.py

- Generate sample pagerank score data
  - GPR.txt
  - QTSPR-U2Q1.txt
  - PTSPR-U2Q1.txt

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

- Most of the dataset is consist of numpy.array format.
- In the case of transition matrix, scipy sparse matrix format is used.
- Use the efficient computation method which is described in the class.
  - For example, when row-sum-zero substitute to 1/n, instead of putting into transition matrix, independent column vector is used
- To minimize the computation, structured numpy is preferred rather than dictionary

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

- requirement.txt is attached.
- Mainly used numpy / scipy / sklearn

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

- Well-performed on my laptop, but the computation time is problem. Most of the data structure is callable as structured or dictionary.
- Some bottlenecks are existing. Improvement is needed.

- Development Environment
  - CPU: i7-8650U @ 1.90GHz
  - Ram: 8Gb
  - OS: Windows 10 / ubuntu on windows 10
  - IDE: pycharm, jupyter notebook, jupyter lab
  - Python 3.6.7

Appendix. MAP / Precision results of 9 approaches

Name	MAP	Pre00	Pre01	Pre02	Pre03	Pre04	Pre05	Pre06	Pre07	Pre08	Pre09	Pre10
Baseline	0.2635	0.8405	0.5926	0.4732	0.3781	0.3145	0.2426	0.1673	0.0915	0.0551	0.0388	0.0101
NS_random	0.0519	0.2179	0.1070	0.0818	0.0758	0.0686	0.0637	0.0512	0.0283	0.0113	0.0074	0.0041
GPR_NS	0.0454	0.1440	0.0873	0.0781	0.0731	0.0695	0.0650	0.0531	0.0300	0.0115	0.0074	0.0041
GPR_WS	0.2588	0.8099	0.5835	0.4726	0.3760	0.3111	0.2418	0.1658	0.0915	0.0573	0.0388	0.0102
GPR_CM	0.2637	0.8405	0.5926	0.4732	0.3782	0.3146	0.2435	0.1682	0.0915	0.0550	0.0388	0.0101
QTSPR_NS	0.0433	0.1367	0.0780	0.0729	0.0703	0.0661	0.0615	0.0498	0.0276	0.0115	0.0073	0.0040
QTSPR_WS	0.2559	0.7881	0.5706	0.4669	0.3740	0.3092	0.2407	0.1662	0.0914	0.0568	0.0392	0.0101
QTSPR_CM	0.2637	0.8405	0.5926	0.4731	0.3781	0.3146	0.2435	0.1682	0.0915	0.0550	0.0388	0.0101
PTSPR_NS	0.0455	0.1479	0.0817	0.0768	0.0731	0.0688	0.0639	0.0492	0.0273	0.0114	0.0073	0.0040
PTSPR_WS	0.2599	0.8124	0.5870	0.4710	0.3780	0.3128	0.2409	0.1671	0.0912	0.0569	0.0392	0.0102
PTSPR_CM	0.2637	0.8405	0.5926	0.4731	0.3780	0.3148	0.2430	0.1677	0.0915	0.0550	0.0388	0.0101

Appendix. Codes for custom method example

```
if CM_method == 'C00':
    combined_coeff = np.random.dirichlet(abs(search_relevance_score_normalized)) # C00 # Random
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C01':
    combined_coeff = abs(abs(pagerank_score_normalized)+abs(search_relevance_score_normalized)) # C01
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C02':
    combined_coeff = abs(abs(pagerank_score_normalized)-abs(search_relevance_score_normalized)) # C02
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C03':
    combined_coeff = sp.mean([abs(pagerank_score_normalized), abs(search_relevance_score_normalized)], axis=0) # C03
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C04':
    combined_coeff = np.amax([abs(pagerank_score_normalized), abs(search_relevance_score_normalized)], axis=0) # C04
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C05':
    combined_coeff = sp.stats.gmean([abs(pagerank_score_normalized), abs(search_relevance_score_normalized)], axis=0) # C05
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C06':
    combined_coeff = sp.stats.hmean([abs(pagerank_score_normalized), abs(search_relevance_score_normalized)], axis=0) # C06
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C07':
    combined_coeff = - np.exp(abs(pagerank_score_normalized)) + np.exp(abs(search_relevance_score_normalized)) # C07
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C08':
    combined_coeff = + np.exp(abs(pagerank_score_normalized)) + np.exp(abs(search_relevance_score_normalized)) # C08
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C09':
    combined_coeff = - np.log(abs(pagerank_score_normalized)) + np.log(abs(search_relevance_score_normalized)) # C09
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
elif CM_method == 'C10':
    combined_coeff = - np.log(abs(pagerank_score_normalized)) - np.log(abs(search_relevance_score_normalized)) # C10
    combined_coeff = normalize(combined_coeff.reshape(1, -1), norm='l1').reshape(-1)
```

Reference

1. Richardson, M., & Domingos, P. (2002). The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Advances in neural information processing systems* (pp. 1441-1448).

2. DeWitt, Y. W. D. J. (2004, October). Computing pagerank in a distributed internet search system. In *Proceedings of the 30th VLDB Conference, Toronto, Canada* (Vol. 30, pp. 420-431).

3. Liu, Fang, Clement Yu, and Weiyi Meng. "Personalized web search for improving retrieval effectiveness." *IEEE Transactions on knowledge and data engineering* 16, no. 1 (2004): 28-40.

4. Kollias, G., Gallopoulos, E., & Szyld, D. B. (2006). Asynchronous iterative computations with Web information retrieval structures: The PageRank case. *arXiv preprint cs/0606047*.

5. Yue, Z., Harpale, A., He, D., Grady, J., Lin, Y., Walker, J., ... & Yang, Y. (2009, July). CiteEval for evaluating personalized social web search. In *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation* (pp. 23-24).

6. Zhou, L. (2015). Personalized Web Search. *arXiv preprint arXiv:1502.01057*.

7. Fidel, R., Bruce, H., Pejtersen, A. M., Dumais, S., Grudin, J., & Poltrock, S. (2016). Collaborative information retrieval (CIR). *The New Review of Information Behaviour Research*, *1*.